# Stabilization Algorithms for Large-Scale Problems

Toke Koldborg Jensen

Kongens Lyngby 2006
IMM-PHD-2006-163

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-PHD: ISSN 0909-3192

# Preface

This thesis is written as a part of the requirements for obtaining the PhD degree at the Technical University of Denmark (DTU). The study was carried out between April 2003 and March 2006 in the scientific computing section at Informatics and Mathematical Modelling (IMM) under supervision of Per Christian Hansen. The project is a part of the PhD programme in mathematics, physics and informatics.

**Acknowledgements**

Many people have influenced the development of this thesis in one way or another. First, I want to thank my supervisor Per Christian Hansen for three years of insightful discussions and valuable support. Likewise, I want to thank James G. Nagy and all the people in the Computer Science group at Emory University – not least all my fellow students in the project room. I would also like to thank Giuseppe Rodriguez and his colleges at the Department of Mathematics and Informatics, Università di Cagliari for their rewarding collaboration and extraordinary hospitality. Back at IMM, I would like to thank all the people in the scientific computing group as well as a lot of other people at IMM – especially all my fellow students and friends who have provided a nice and productive atmosphere. I would like also to thank MSc Jesper Pedersen for proofreading this thesis, and for a lot of creative discussions during the last several years.

Finally, I would like to thank my family and friends, and especially my girlfriend Mie, who have accomplished to provided a relaxed atmosphere (with almost no inverse problems) when I most needed it.

Kgs. Lyngby, March 31, 2006

Toke Koldborg Jensen

# Resumé

Projektets fokus ligger på stabilisering af storskalaproblemer hvor strukturerede modeller og iterative algoritmer er nødvendige for at kunne beregne tilnærmede løsninger. Afhandlingen omfatter derfor et studie af forskellige iterative Krylov-metoder og disses anvendelse på inverse problemer. Nogle af metoderne er tidligere identificeret som metoder, der kan generere regulariserede løsninger, hvorimod andre har været foreslået i literaturen, men kun sparsomt studeret i praksis. Dette projekt bidrager på afgørende vis til forståelsen af disse metoder.

Billedrefokuseringsproblemer er et godt eksempel på en klasse af storskalaproblemer for hvilken de forskellige metoder kan afprøves. Derfor har denne klasse af problemer givet anledning til et selvstændigt studie af de forskellige matrixstrukturer, der optræder i denne forbindelse – ikke mindst for at skabe et fælles grundlag for diskussioner.

Et andet vigtigt aspekt er regulariseringsmatricer til formulering af inverse problemer på generel form. Specielle klasser af regulariseringsmatricer for storskalaproblemer (herunder todimensionelle problemer) er analyseret. Desuden er ovennævnte Krylov-metoder også analyseret i forbindelse med løsning af problemer på generel form og en udvidelse til metoderne er udviklet til dette formål.

L-kurver udgør en af flere parametervalgsmetoder, der kan anvendes i forbindelse med løsning af inverse problemer. I forbindelse med projektet har en del af arbejdet resulteret i en ny heuristik til at lokalisere hjørnet af en diskret L-kurve. Denne heuristik er implementeret som en del af en større algoritme, der er udviklet i samarbejde med G. Rodriguez og P. C. Hansen.

Sidst, men ikke mindst, har en stor del af projektet på forskellig vis omhandlet den objekt-orienterede MATLAB toolbox MOORe Tools, der er udviklet af PhD Michael Jacobsen. Nye implementeringer er tilføjet og flere fejl og mangler er udbedret.

Projektet har resulteret i udarbejdelsen af tre artikler, der alle for nemhedens skyld er inkluderet i et appendix.

# Abstract

The focus of the project is on stabilization of large-scale inverse problems where structured models and iterative algorithms are necessary for computing approximate solutions. For this purpose, we study various iterative Krylov methods and their abilities to produce regularized solutions. Some of the Krylov methods have previously been studied and identified as iterative regularization methods, whereas others have been proposed in the literature, but only sparsely studied in practise. This thesis considerably improves the understanding of these methods.

Image deblurring problems constitute a nice class of large-scale problems for which the various methods can be tested. Therefore, this present work includes a separate study of the matrix structures that appear in this connection – not least to create a common basis for discussions.

Another important part of the thesis is regularization matrices for the formulation of inverse problems on general form. Special classes of regularization matrices for large-scale problems (among these also two-dimensional problems) have been analyzed. Moreover, the above mentioned Krylov methods have also been analyzed in connection with the solution of problems on general form, and a new extension to the methods has been developed for this purpose.

The L-curve method is one among several parameter choice methods that can be used in connection with the solution of inverse problems. A part of the work has resulted in a new heuristic for the localization of the corner of a discrete L-curve. This heuristic is implemented as a part of a larger algorithm which is developed in collaboration with G. Rodriguez and P. C. Hansen.

Last, but not least, a large part of the project has, in different ways, revolved around the object-oriented MATLAB toolbox M$\mathcal{OO}$Re Tools developed by PhD Michael Jacobsen. New implementations have been added, and several bugs and shortcomings have been fixed.

The work has resulted in three papers that are all included in an appendix for convenience.

# Symbol List

The thesis deals with several different topics, and therefore a lot of notation is needed. Here, we provide some general remarks as well as a list of commonly used notation.

In general, matrices are denoted by uppercase letters such as $A$ or $\Sigma$. If the matrix $A$ is full, then $a_i$ denotes the $i$th column of $A$, whereas if $\Sigma$ is a diagonal matrix, then $\sigma_i$ denotes the $i$th diagonal element. A single element of a full matrix is denoted as $a_{i,j}$. Moreover, $V_k$ denotes a full matrix with $k$ columns.

Vectors are generally denoted by lowercase letters such as $x$ and $b$. Note that also lowercase Greek letters such as $\beta$ and $\xi$ are vectors in certain contexts. A specific element of a vector is denoted with a subscript, e.g., $x_i$.

Calligraphy letters are mainly used to denote operators in the continuous domain, as well as polynomials. For example, $\mathcal{K}$ is a compact linear operator, and $\mathcal{P}_k$ is a polynomial of degree $\leq k$.

| Symbol | Description |
|---|---|
| $\mathcal{K}$ | Compact linear operator |
| $f(t)$ | Solution to continuous problem |
| $g(s)$ | Right-hand side to continuous problem |
| $A$ | Coefficient matrix / PSF matrix |
| $b$ | Right-hand side vector |
| $b^\delta$ | Noisy right-hand side vector |
| $e$ | Vector of white Gaussian noise |
| $L$ | General regularization matrix for the general-form problem |
| $x_\alpha$ | Regularized solution due to the regularization parameter $\alpha$ |
| $x_{L,\alpha}$ | Regularized general-form solution due to the regularization matrix $L$ and the regularization parameter $\alpha$ |
| $\varepsilon_2(x_\alpha)$ | Relative error of the regularized solution $x_\alpha$ compared to the true solution (measured in the 2-norm) |
| $x^{(k)}$ | The $k$th iterate of GMRES or MINRES |

| | |
|---|---|
| $\bar{x}^{(k)}$ | The $k$th iterate of CGLS/LSQR |
| $\hat{x}^{(k)}$ | The $k$th iterate of RRGMRES or MR-II |
| $\mathcal{P}_k$, $\mathcal{Q}_k$ | Solution and residual polynomials for GMRES or MINRES |
| $\overline{\mathcal{P}}_k$, $\overline{\mathcal{Q}}_k$ | Solution and residual polynomials for CGLS/LSQR |
| $\widehat{\mathcal{P}}_{k+1}$, $\widehat{\mathcal{Q}}_{k+1}$ | Solution and residual polynomials for RRGMRES or MR-II |
| $H_k$ | Hessenberg matrix resulting from Arnoldi's method |
| $B_k$ | Bidiagonal matrix resulting from the Lanczos bidiagonalization method |
| $W_k$ | Matrix with columns that span $\mathcal{K}_k(A, b)$ |
| $\overline{W}_k$ | Matrix with columns that span $\mathcal{K}_k(A^TA, A^Tb)$ |
| $\widehat{W}_k$ | Matrix with columns that span $\mathcal{K}_k(A, Ab)$ |
| $\alpha$ | Denotes some unspecified regularization parameter |
| $\lambda$ | The Tikhonov regularization parameter |
| $U\Sigma V^T$ | Denotes the SVD of the coefficient matrix $A$ |
| $\nu$ | Parameter for the stopping rule defined in Definition 4.8 |
| $^{\{\cdot\}}$ | Superscript curly brackets contain type of derivative operator and/or type of boundary conditions |
| $\mathcal{N}(\cdot)$ | The nullspace of a matrix |
| $\mathcal{R}(\cdot)$ | The range of a matrix |
| $L^\dagger$ | The Moore-Penrose pseudoinverse of $L$ |
| $L_A^\dagger$ | The $A$-weighted pseudoinverse of $L$ |
| $\Pi_k$ | The $k$th point on a discrete L-curve |
| $\otimes$ | Kronecker product |

# Contents

Chapter 1

# Introduction

We start this thesis with a small exercise adopted from anvari.org [1] that in a straightforward way describes the essence of an inverse problem. Consider the following Jeopardy like question:

"To what question is the answer "9W.""

To pose the right question to a given answer is often not an easy task. It is especially difficult to guess the right answer if no background knowledge is available. A quick internet search for the string "9W" shows that the answer could be connected to the US highway infrastructure where the road US-9W ends in Albany. Or the answer could involve the "Paul Rodgers/9W" gallery in New York. Knowing that the background for the question is a discussion with a German professor narrows down the possible questions, though the correct question is still not easily found. For convenience, we give here the right question: "Dr. Wiener, do you spell your name with a V?"

Nevertheless, the field of *inverse problems* deals with posing the right questions to known answers – or put in another way – reconstructing interior or hidden premises for an outer observation. In the physical world, questions and answers are posed in terms of models, input parameters, and output values. In a forward problem, given a model and a set of parameters, one can evaluate the model and compute the actions. An example of an advanced modeling framework is the generation of weather forecasts. Using a series of measurements and a model that describes the weather systems, one tries to calculate the weather of tomorrow. While forward modelling indeed can be troublesome, then imagine going the other way. At first it

INPUT (answer) + MODEL $\longrightarrow$ OUTPUT (question)

Figure 1.1: Schematical illustration of a general inverse problem

might seem like an absurd project of no relevance, but what if we want to estimate the development of the global temperature of the Earth over the last 3 mio. years. This can definitely only be done indirectly and by a serious backtracking.

A generic problem can be described in general as in Fig. 1.1. Here the input is the true solution that we later want to estimate, the model is a description of how the input is distorted – how the answer is posed like a question – and the output is the given question.

Indeed there are many practical problems that fall into the category of inverse problems, e.g., when geophysicists want to track the activity of a volcano by looking at the distribution of magnetization deep below the surface [4]. Digging down might be a dangerous task, and it is in practice often impossible. But one might measure the magnetic field above the surface, and from this try to estimate the distribution of magnetization deep below the surface; i.e., estimate interior properties through exterior measurements. Many other areas such as image deblurring, tomography scanning, signal processing, etc., are also areas where inverse problems arise naturally.

Let us turn towards the mathematics and express the generic model from Fig. 1.1 in mathematical terms. Very often this kind of problem can be formulated as a singular integral equations, e.g., as a Fredholm integral equation of the first kind

$$\int_0^1 K(s,t)f(t)\,dt = g(s), \quad 0 \le s \le 1, \tag{1.1}$$

where $K(s,t)$ is a kernel that describes the system, $g(s)$ is the measured signal, and $f(t)$ is the unknown internal solution. In the above formulation the problem is normalized such that $t,\ s\ \in [0;1]$. The difficulties are here reflected in the kernel $K(s,t)$ which is often rank-deficient or ill-conditioned. Basically, this means that some information present in $f(t)$ is lost or damped severely when evaluating the integral. Consequently, when we want estimate $f(t)$ from the measurements in $g(s)$, then we either lack information, or have available some information that is severely damped and possibly noisy. The computed solutions are therefore likely to be severely contaminated by inverted noise.

Very often, an analytical solution to (1.1) is not available, and we need to compute solutions numerically on a computer. There are several ways to discretize (1.1), and we end up with a simple discrete formulation of the following kind

$$Ax = b, \tag{1.2}$$

where $A$ is a matrix that describes a discretization of the kernel $K(s,t)$, $b$ is a discrete vector that represents $g(s)$, and $x$ is a solution vector that describes the wanted solution function $f(t)$. (Note that we study only linear inverse problems.) This formulation can be used in connection with the numerical methods discussed in this present

thesis — and despite its apparent simplicity, a rank-deficiency or ill-conditioning of the physical problem carries over to the discrete domain and results in huge condition numbers. Solving a problem of this kind is definitely not straightforward.

The aim of the present thesis is to treat different aspects of large-scale inverse problems in a general framework, and in this connection investigate the regularization properties of some state-of-the-art iterative algorithms. The MATLAB toolbox M$\mathcal{OO}$Re Tools developed by PhD Michael Jacobsen is widely used, and a secondary aim is to maintain and further develop and debug this object-oriented package.

## 1.1 Outline of the Thesis

The present thesis is written as a self-contained document that describes the investigations and results of the PhD project that has been carried out. At the time of writing, the work has resulted in two accepted journal papers; one with Hansen [40], and one with Hansen and Rodriguez [41]. A third paper [53], also with Hansen, has been submitted. Despite some overlap between some chapters of the thesis and these papers, several additional comments, analyses, and examples are given in the thesis. Furthermore, the thesis includes both a chapter and appendices that describe details involved with the implementations in the M$\mathcal{OO}$Re Tools framework. The rest of the thesis is organized as follows:

Chapter 2 introduces regularization in a mathematical framework and gives the basic background for some standard regularization methods.

Chapter 3 investigates some special properties for two-dimensional image reconstruction problems. These include blurring models, boundary conditions, as well as the corresponding matrix structures.

In Chapter 4 we turn to look at Krylov subspace methods and their ability to provide regularized solutions. It is well-known (see, e.g., [31, 33, 73]) that conjugate gradient type methods applied to the least squares problem have a regularizing effect. But recently, also other minimum-residual methods have been proposed as regularization methods. These methods are studied, and a number of examples illustrate their abilities. Some of the results and examples from this chapter are submitted to BIT [53].

Chapter 5 deals with regularization in general form. Especially, regularization matrices for problems of more than one dimension are described. The chapter includes several results about the implementations of such multidimensional regularization matrices.

In Chapter 6 we return to look at Krylov methods used as iterative regularization methods, but now in a general-form framework. Again, conjugate gradient type methods have previously been used to compute general-form solutions by means of implicit standard-form transformations [33]. The question is whether this carries over to more general minimum-residual methods. A paper that describes a smoothing-norm preconditioner for minimum-residual methods have been accepted for publication [40].

Chapter 7 deals with another part of the history, namely selection of the regularization parameter, specifically finding the corner of a discrete L-curve. The background for this work is a collaboration with Giuseppe Rodriguez from Universitá di Cagliari, and the main algorithm has been published [41]. The focus in this chapter is different than in the paper and more emphasis is put on the basic idea of one part of the published algorithm.

Chapter 8 revolves around some of the new implementations in M$\mathcal{OO}$Re Tools including new classes, utility routines, and extensions to algorithms. New functionality is not always (in fact seldom) easy to implement, but some of these problems are addressed in the appendices.

The conclusion and ideas for future work appear in Chapter 9.

There are three appendices. The first two describe implementation issues that are not strictly relevant for the thesis. Nevertheless, many implementation details of the kind described in the appendices have taken up a large amount of time during the project. Most implementations for the thesis are done in the M$\mathcal{OO}$Re Tools framework, and as the appendices illustrate, the M$\mathcal{OO}$Re Tools toolbox had (and possibly still has) a number of weaknesses and errors. The work with the toolbox has improved a lot of things and several issues have been corrected.

For convenience, the third appendix includes the latest versions of the three papers that have been produced.

Chapter 2

# Regularization

The introduction indicates that inverse problems can be very hard to solve. In fact, to approximately solve the problems, we need to enforce some kind of *regularity* on the solution, and in general regularization aims at applying additional restrictions to the solutions than the system itself provides; e.g., a standard approach is to restrict the "size" of the solution measured in some appropriate norm.

Apart from the simpler norm constraints, one might also want a solution that is nonnegative, or obeys some monotonicity constraints. Also, certain elements of the solution can be known to have specific values.

With such restrictions, we are hopefully able to compute solutions that approximate the underlying and unknown true solution, and these solutions are hopefully better than naively computed solutions to the ill-conditioned system.

This chapter introduces the basic notation, some general concepts, as well as standard mathematical tools for analyzing and describing these kinds of problems.

## 2.1 The Continuous Problem

We start by looking at the continuous Fredholm integral equation of first kind (1.1). In more abstract terms, the original problem can also be stated as an operator equation through the compact linear integral operator $\mathcal{K} : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X}$ and $\mathcal{Y}$ are Hilbert spaces. That is., the operator $\mathcal{K}$ is defined as $\mathcal{K}f(s) = \int_0^1 K(s,t)f(t)\,dt$ such that Eq. 1.1 can be written as

$$\mathcal{K}f = g, \tag{2.1}$$

where again $\mathcal{K}$ represents the model, $g$ the observations, and $f$ the sought solution.

Hadamard was the first to describe the ill-posed nature of some problems, and he believed that these problems could have no physical meaning. Obviously, he was wrong. He set up three criteria for a problem to be well-posed – and violation of any of these criteria will lead to an ill-posed problem.

**Definition 2.1** (Hadamard's Criteria for a Well-Posed Problem)

1. For any data, a solution exists. (Existence)
2. The solution is unique. (Uniqueness)
3. The solution depends continuously on the data. (Stability)

While the violation of either of the first two statements is indeed serious, it is often the stability issue that causes the most trouble, especially when computing solutions numerically. The first two statements are connected to the attainability of the right-hand side and whether or not the linear operator $\mathcal{K}$ has a nontrivial nullspace. To deal with these problems, we can use some *generalized inverse* of $\mathcal{K}$ for computing solutions that are in some sense optimal. For practical discrete and noisy systems, the stability problem means that the solutions must be further stabilized. We refer to especially [22], but also [29, 37] and references therein for more in-depth analyses of these issues. The following derivations are also, in part, based on [22].

The trouble with the stability begins because the inverse of a compact linear operator $\mathcal{K}$ is unbounded. In vague terms, this means that even a tiny perturbation of the right-hand side function, $g^\delta = g + d$, where $\|d\|/\|g\|$ is small, can have a disastrous effect on the solution $f^\delta = \mathcal{K}^{-1}g^\delta$.

Any compact linear operator $\mathcal{K}$ has a singular system that consists of the non-increasing singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$, and the corresponding orthonormal singular functions $v_i(t)$ and $u_i(s)$:

**Definition 2.2** (SVE) The singular value expansion of the compact linear operator $\mathcal{K}$ is defined by

$$\mathcal{K} = \sum_{i=1}^{\infty} \sigma_i u_i(s) v_i(t),$$

where $u_i(s)$ and $v_i(t)$ are orthonormal functions called the left and right singular functions, and $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$ are non-negative singular values.

We express the functions $f(t)$ and $g(s)$ in terms of the left and right singular functions such that

$$f = \sum_{i=1}^{\infty} \langle f,\, v_i \rangle v_i \quad \text{and} \quad g = \sum_{i=1}^{\infty} \langle g,\, u_i \rangle u_i, \tag{2.2}$$

where $\langle \cdot,\, \cdot \rangle$ denotes the usual inner product. It follows from Definition 2.2 that for the linear operator $\mathcal{K}$, we have

$$\mathcal{K} v_i = \sigma_i u_i \quad \text{and} \quad \mathcal{K}^* u_i = \sigma_i v_i$$

where $\mathcal{K}^*$ is the conjugate operator, and thus

$$\mathcal{K}f = \sum_{i=1}^{\infty} \sigma_i \langle f,\, v_i \rangle u_i \quad \text{and} \quad \mathcal{K}^* g = \sum_{i=1}^{\infty} \sigma_i \langle g,\, u_i \rangle v_i.$$

To avoid the possible problems with the existence and uniqueness of the solutions to (2.1), we want to compute the minimum-norm least squares solution. To do this, we must avoid any components from the possible nullspace of $\mathcal{K}$. Similarly, any component of $g$ that does not lie in $\mathcal{R}(\mathcal{K})$, i.e., lying in the nullspace of $\mathcal{K}^*$, will not affect the least squares problem. Thus, the minimum-norm least squares solution is the unique solution of $\mathcal{K}P_{\mathcal{R}(\mathcal{K}^*)} f_{\text{ls}} = P_{\mathcal{R}(\mathcal{K})} g$ where $P_{\mathcal{R}(\mathcal{K}^*)}$ and $P_{\mathcal{R}(\mathcal{K})}$ are projectors onto $\mathcal{R}(\mathcal{K}^*)$ and $\mathcal{R}(\mathcal{K})$, respectively. Using the decompositions of $f$ and $g$ from (2.2), we get the projected problem

$$\sum_{\sigma_i > 0} \sigma_i \langle f,\, v_i \rangle u_i(s) = \sum_{\sigma_i > 0} \langle g,\, u_i \rangle u_i(s),$$

and we can express the minimum-norm least squares solution to (1.1) in terms of the SVE as

$$f_{\text{ls}}(t) = \mathcal{K}^\dagger g(s) = \sum_{\sigma_i > 0} \frac{\langle g,\, u_i \rangle}{\sigma_i} v_i(t),$$

by summing up all the orthogonal solution components that correspond to nonzero singular values. We definitely want the solution to be finite, i.e., a requirement for a solution to exist is that

$$\|\mathcal{K}^\dagger g\|_2^2 = \sum_{\sigma_i > 0} \frac{|\langle g,\, u_i \rangle|^2}{\sigma_i^2} < \infty, \tag{2.3}$$

which is called the Picard criterion [35]. Later, we will see how a similar criterion arises for discrete ill-posed problems, and how Picard plots can be used to analyze the discrete problems.

Above, we assume that $g(s)$ is the true right-hand side to the system. Now let $g^\delta(s) = g(s) + d(s)$ be a noisy right-hand side where $d(s)$ is some function that describes the noise. Note that we can split up the solution into a signal component and a noise component

$$\|\mathcal{K}^\dagger g^\delta\|_2^2 = \sum_{\sigma_i > 0} \frac{|\langle g,\, u_i \rangle + \langle d,\, u_i \rangle|^2}{\sigma_i^2},$$

where the noise component most likely implies that the Picard criterion is not fulfilled. In fact, even tiny perturbations $d$ of the right-hand side can completely destroy the solutions due to division by the steadily decreasing singular values leading to serious stability problems.

We therefore need to stabilize the solutions, and we basically have two different approaches to consider. First, if we know a good solution subspace $\mathcal{S}_k$ of some

dimension $k$, then we can restrict the solution to lie in this subspace, avoiding any noisy components from $\mathcal{S}_k^{\perp}$. This approach is generally termed a *projection method*. We can also define an optimization problem by formulating meaningful constraints or penalties on the wanted solution, e.g., that the size of the solutions should be bounded (such that the noise is not allowed to blow up too much)

$$\min_f \|\mathcal{K}f - g^\delta\|_2^2 \quad \text{s.t.} \quad \|f\|_2^2 \leq \alpha,$$

where we minimize the residual while keeping the solution norm below some threshold $\alpha$. These approaches are generally termed *penalty methods.* From the optimization theory we know that the above constrained problem can be reformulated to yield the unconstrained problem

$$\min_f \left\{ \|\mathcal{K}f - g^\delta\|_2^2 + \lambda^2 \|f\|_2^2 \right\},$$

where $\lambda^2$ is a Lagrangian multiplier connected to the threshold $\alpha$. This formulation is known as the Tikhonov problem due to A. N. Tikhonov [89], and it is obvious that we try to balance on one hand the fit of the solution, and on the other hand the size of the solution.

In a more general framework we can define the concept of a regularized operator or a regularized inverse. We know that the least squares solution of minimum norm $\mathcal{K}^\dagger g^\delta$ is often meaningless because of the unboundedness of $\mathcal{K}^\dagger$ and the noise in $g^\delta$, but we still want to compute the parts of the solution that are meaningful. We therefore define the regularized operator $\mathcal{R}_\alpha$ depending on the parameter $\alpha$, and replace the unbounded inverse $\mathcal{K}^\dagger$ with the continuous inverse approximation $\mathcal{R}_\alpha$ such that a regularized solution is given by $f_\alpha^\delta = \mathcal{R}_\alpha g^\delta$.

Until now we assumed that some parameter $\alpha$ can be found such that $\mathcal{R}_\alpha$ is a suitable regularized operator. Thus when defining a *regularization method*, one has to consider both a regularized operator, as well as a way of choosing the regularization parameter – i.e., a parameter choice method is required. Furthermore, a standard requirement for the regularized operator $\mathcal{R}_\alpha$ is that when the noise level $\delta$ goes to zero then $\mathcal{R}_\alpha \to \mathcal{R}_0 = \mathcal{K}^\dagger$, i.e., the parameter choice method should give $\alpha = 0$ and the regularized operator should approach the Moore-Penrose pseudoinverse $\mathcal{K}^\dagger$ and thus give the minimum-norm least squares solution of the undisturbed problem. From the above, and inspired by Engl et al. [22, Definition 3.1] we define

**Definition 2.3** (Regularized operator) Let $\mathcal{K} : \mathcal{X} \to \mathcal{Y}$ be a bounded linear operator between the Hilbert spaces $\mathcal{X}$ and $\mathcal{Y}$, and let $\alpha = \alpha(\delta, g^\delta) \in [0, \infty[$ be a parameter chosen by a parameter choice method based on the noise and the perturbed right-hand side. Then let

$$\mathcal{R}_\alpha \, : \, \mathcal{Y} \to \mathcal{X}$$

be a continuous (not necessarily linear) operator. The family of operators $\{\mathcal{R}_\alpha\}$ is called a regularization operator for $\mathcal{K}^\dagger$ if, for all $g \in \mathcal{D}(\mathcal{K}^\dagger)$ it holds that

$$\lim_{\delta \to 0} \sup \left\{ \|\mathcal{R}_\alpha g^\delta - \mathcal{K}^\dagger g\| \quad | \quad g^\delta \in \mathcal{Y}, \, \|g^\delta - g\| \leq \delta \right\} = 0.$$

Furthermore, if the parameter choice method fulfills

$$\lim_{\delta\to 0} \sup\left\{\alpha(\delta, g^\delta) \quad | \quad g \in \mathcal{Y},\, \|g^\delta - g\| \le \delta\right\} = 0,$$

then the pair $(\mathcal{R}_\alpha, \alpha)$ is called a regularization method.

It is seen from this formal definition that knowledge about the noise level is needed to define a regularization method. Particularly, [22, Theorem 3.3] shows that no error-free parameter choice method can yield a convergent regularization method. But in several practical applications we are interested in finding approximate solutions anyway, and the general definition serves merely as a mental guideline. We will go more into detail when discussing the discretized problems next.

## 2.2 The Discrete Problem

Some problems can be solved analytically using the continuous formulation. But often we need numerical solution techniques, and we need to discretize the problems. That is, we need to express the functions $f(t)$ and $g(s)$ from (1.1) by vectors, and express the effect of the compact operator $\mathcal{K}$ according to the discretized functions as a discrete operator, e.g., as a matrix. We want to obtain a discrete system of linear equations of the form

$$Ax = b, \quad A \in \mathbb{R}^{m\times n}, \quad x \in \mathbb{R}^n, \quad b \in \mathbb{R}^m, \tag{2.4}$$

where $A$ is a discrete representation of $\mathcal{K}$, and $x$ and $b$ are discrete representations of $f$ and $g$, respectively. There are several choices for arriving at a discrete system of equations, see e.g., [2]. Two main approaches are the following.

**Quadrature Methods.** The function $f(t)$ can be discretized by sampling the function values in a number of discrete points $t_j$, $j = 1, 2, \ldots, n$, thus $x_j = f(t_j)$ and $x \in \mathbb{R}^n$ is a vector which contains the sampled values. The integration over $t$ can now be accomplished by applying some quadrature rule to the vector $x$, i.e., given the quadrature weights $w_j$ the discretized integral is given by

$$\int_0^1 K(s,t)f(t)\,dt \approx \sum_{i=1}^{n} w_j K(s,t_j)x_j.$$

An example of a simple quadrature method is the trapezoidal rule where the integral is approximated by linearly connecting two successive values $x_j$ and $x_{j+1}$ for $j = 1, \ldots, n-1$. Using equidistant grid spacing $h = t_{j+1} - t_j$, the resulting weights are given by

$$w_j = \begin{cases} 0.5h^{-1} & \text{for} \quad j = 1,\, n \\ h^{-1} & \text{for} \quad j = 2,\, 3,\, \ldots,\, n-1 \end{cases}$$

Now sample the right-hand side function $g(s)$ in $m$ distinct points $s_i$ for $i = 1, 2, \ldots, m$ and apply the above quadrature rule for each point to get a discrete system of linear equations (2.4) where the elements of $A$ are $a_{ij} = w_j K(s_i, t_j)$.

**Galerkin Methods.** Using this method, we first define two sets of basis functions $\theta_j(t)$ and $\phi_i(s)$ and expand an approximation to the wanted solution $f(t)$ in terms of $\theta_j(t)$

$$f(t) \approx \tilde{f}(t) = \sum_{j=1}^{n} x_j \theta_j(t).$$

We then insert this into the integral equation, and obtain

$$\int_0^1 K(s,t) \sum_{j=1}^{n} x_j \theta_j(t)\, dt = g(s).$$

By making sure that the residual is orthogonal to each of the basis functions $\phi_j$, i.e., $\int_0^1 \phi_i(s) \left( \int_0^1 K(s,t) \sum_{j=1}^{n} x_j \theta_j(t)\, dt - g(s) \right) ds = 0$, we obtain for each basis vector $\phi_i$

$$\int_0^1 \int_0^1 K(s,t) \phi_i(s) \sum_{j=1}^{n} x_j \theta_i(t)\, dt\, ds = \int_0^1 g(s) \phi_i(s)\, ds.$$

The vector $x \in \mathbb{R}^n$ now contains the coefficients to each of the $n$ basis functions $\theta_j$ for $j = 1, \ldots, n$, and similarly the elements of the right-hand side $b \in \mathbb{R}^m$ are given by $b_i = \int_0^1 g(s) \phi_i(s)\, ds$. The elements of the coefficient matrix $A \in \mathbb{R}^{m \times n}$ are given by $a_{ij} = \int_0^1 \int_0^1 K(s,t) \phi_i(s) \theta_j(t)\, dt\, ds$, and again we obtain a linear system of the form (2.4).

The discretization is by itself a projection from the continuous domain to a finite dimensional discrete domain, e.g., a projection onto the basis spanned by the discretized functions $\phi_i$. If the discretization is very coarse then the discrete subspace might avoid the troublesome parts of the original continuous kernel, and therefore yield a regularized solution. But often, the projected discrete system mimics the ill-posed properties of the continuous formulation which shows up as ill-conditioning of the system matrix $A$.

We therefore need to apply some stabilization algorithm, also for solving the discrete system $Ax = b$ (or $\min_x \|b - Ax\|_2$ in case the system is inconsistent). That is, we want either to find a subspace of the discrete domain that is a good solution subspace (projection method) or to formulate a constrained optimization problem (penalty method), in shape of the Tikhonov problem in discrete form.

Similarly to the continuous setting, we rarely have the true right-hand side $b$, but rather some measured right-hand side $b^\delta = b + e$ where $e$ denotes a noise vector

that includes measurement noise, discretization errors, etc. For practical problems the noise is often bandlimited and can have any distribution. Moreover, the noise does not need to be only additive. But for simplicity, we here, and throughout the thesis use the standard text-book assumption that the noise is only additive, and that it is white and follows a Gaussian distribution. The parameter $\delta$ indicates the signal-to-noise level, and we define the noise level as

$$\delta = \|e\|_2 / \|b\|_2. \tag{2.5}$$

We therefore often do not solve the noise-free problem (2.4), but instead

$$Ax = b^\delta, \quad \text{or} \quad \min_x \|b^\delta - Ax\|_2. \tag{2.6}$$

where the least squares problem is solved in case the system is inconsistent, i.e., $b^\delta \notin \mathcal{R}(A)$. In this case the linear system has no solution, but the least squares problem does.

### 2.2.1 Matrix Decomposition

In the continuous case, the SVE was used to expand the solution and the right-hand side in the singular functions. This expansion was in turn used to describe the ill-conditioning through the Picard criterion. To analyze the discrete case, we start by introducing a tool similar to the SVE, namely the singular value decomposition.

**Definition 2.4** (SVD) Let $A \in \mathbb{R}^{m\times n}$ be a matrix and let $m \geq n$. Then the singular value decomposition (SVD) of $A$ is defined by

$$A = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^T,$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal, and $\Sigma \in \mathbb{R}^{n\times n}$ is a diagonal matrix that contains the nonnegative singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. In case $m < n$ then define the SVD from $A^T$.

Now $A$, the solution $x$, and the noisy right-hand side $b^\delta = b + e$ can be expressed in the singular vector bases $U = (\, u_1,\, u_2,\, \ldots,\, u_m\,)$ and $V = (\, v_1,\, v_2,\, \ldots,\, v_n\,)$ as

$$A = \sum_{i=1}^{n} \sigma_i u_i v_i^T, \quad x = \sum_{i=1}^{n} (v_i^T x) v_i, \quad b^\delta = \sum_{i=1}^{m} (u_i^T b^\delta) u_i. \tag{2.7}$$

Let $r = \operatorname{rank}(A) \leq n$ be the rank of $A$, then we can always write the Moore-Penrose pseudoinverse $A^\dagger$ in terms of the SVD as

$$A^\dagger = \sum_{i=1}^{r} \sigma_i^{-1} v_i u_i^T.$$

The Moore-Penrose pseudoinverse fulfills the four Moore-Penrose conditions, see [26, §5.5.4] and the definition below.

**Definition 2.5** Let $A \in \mathbb{R}^{m\times n}$ be a general matrix, then the Moore-Penrose pseudoinverse $A^\dagger$ of $A$ fulfills the four Moore-Penrose conditions

$$AA^\dagger A = A, \qquad A^\dagger AA^\dagger = A^\dagger$$

$$(AA^\dagger)^T = AA^\dagger, \qquad (A^\dagger A)^T = A^\dagger A,$$

which implies that $AA^\dagger$ and $A^\dagger A$ are orthogonal projections onto $\mathcal{R}(A)$ and $\mathcal{R}(A^T)$, respectively.

If $A$ is square ($m = n$) and has full rank ($\sigma_n > 0$) then $A^\dagger$ is identical to the ordinary inverse $A^{-1}$, which in this case is well-defined. In general, the Moore-Penrose pseudoinverse can be used to express the least squares solution of minimum 2-norm as

$$x_{\text{ls}} = A^\dagger b^\delta = \sum_{i=1}^{r} \frac{u_i^T b^\delta}{\sigma_i} v_i = \sum_{i=1}^{r} \left( \frac{u_i^T b}{\sigma_i} + \frac{u_i^T e}{\sigma_i} \right) v_i, \tag{2.8}$$

and we note that, similarly to the continuous formulation, we can theoretically split up the contributions from the true right-hand side and the additive noise. Similar to the Picard criterion (2.3), we note that to be able to compute a meaningful solution to a discrete ill-posed problem then the right-hand side components $|u_i^T b^\delta|$ must (at least on average) decay faster than the singular values. This is called the discrete Picard condition [35]. Because of the smoothing properties of $A$ and that the true right-hand side components must fulfill the discrete Picard condition, then the latter singular components are much more likely to be affected by noise than the first. We will later see an example of a so-called Picard plot which shows the right-hand side components $|u_i^T b|$, the singular values $\sigma_i$, and the resulting solution components $|u_i^T b|/\sigma_i$.

### 2.2.2 Truncated SVD and Tikhonov Regularization

We want to filter out the unwanted components, and introduce the two most used regularization methods; *truncated SVD* (TSVD) and *Tikhonov regularization*. The TSVD method arises naturally by truncating the sum in (2.8) and compute the solution based on only a selected number of terms, $k < r$. The TSVD solution is therefore given as

$$x_k = \sum_{i=1}^{k} \frac{u_i^T b^\delta}{\sigma_i} v_i, \tag{2.9}$$

where the subspace $\text{span}\{v_1, v_2, \ldots, v_k\}$ hopefully carries enough and relevant information for the solution $x_k$ to be a good approximation to the true discrete solution $x$ that is potentially spanned by all $n$ basis vectors (2.7). This solution, in turn, is then hopefully a good approximation to the wanted continuous true solution.

The other approach is based on the fact that we want to minimize the residual, but at the same time restrict the norm of the solution. This can be set up in the simplest case as

$$x_\lambda = \text{argmin}_x \left\{ \|b^\delta - Ax\|_2^2 + \lambda^2 \|x\|_2^2 \right\}, \tag{2.10}$$

where the parameter $\lambda$ is the *regularization parameter*. This parameter controls the amount of regularization to impose, and should be selected carefully. By letting $\lambda = 0$, we get the standard least squares solution, and for $\lambda = \infty$, the minimizer is $x = 0$. The Tikhonov problem (2.10) has two other equivalent formulations. By stacking the two norms, we arrive at the first expression below, and by evaluating the norm and finding the minimum through differentiation, we arrive at the latter

$$\min \left\| \begin{pmatrix} A \\ \lambda I \end{pmatrix} x - \begin{pmatrix} b^\delta \\ 0 \end{pmatrix} \right\|_2 \quad \text{and} \quad \left(A^T A + \lambda^2 I\right) x = A^T b^\delta. \tag{2.11}$$

From the last formulation, and by using the SVD of $A$, we can again write the solution as a sum over the singular components that depend on the regularization parameter $\lambda$. Note that we can express both $A^T A$ and $A^T b^\delta$ using the right singular vectors

$$A^T A = \sum_{i=1}^{n} \sigma_i^2 v_i v_i^T \quad \text{and} \quad A^T b = \sum_{i=1}^{n} \sigma_i (u_i^T b^\delta) v_i$$

such that the Tikhonov solution can be formulated as

$$\begin{aligned} x_\lambda &= (A^T A + \lambda^2 I)^{-1} A^T b^\delta = \sum_{i=1}^{n} v_i (\sigma_i^{-2} + \lambda^{-2}) v_i^T \sigma_i (u_i^T b^\delta) v_i \\ &= \sum_{i=1}^{n} \frac{\sigma_i^2}{\sigma_i^2 + \lambda^2} \frac{u_i^T b^\delta}{\sigma_i} v_i. \end{aligned} \tag{2.12}$$

The TSVD and the Tikhonov solutions can therefore both be written as filtered SVD solutions

$$x_\alpha = \sum_{i=1}^{n} \phi_{\alpha,i} \frac{u_i^T b^\delta}{\sigma_i} v_i = V \Phi_\alpha \Sigma^\dagger U^T b \quad , \quad \Phi_\alpha = \text{diag}\left(\phi_{\alpha,1}, \ldots, \phi_{\alpha,n}\right). \tag{2.13}$$

where $x_\alpha$ is a regularized solution, $\alpha$ is some regularization parameter, and the $\phi_{\alpha,i}$s are so-called *filter factors*. The filter factors are given as

$$\phi_{k,i} = \begin{cases} 1 & \text{for } i \leq k \\ 0 & \text{otherwise} \end{cases} \quad , \quad \phi_{\lambda,i} = \frac{\sigma_i^2}{\sigma_i^2 + \lambda^2}, \tag{2.14}$$

for TSVD and Tikhonov regularization, respectively. For $\lambda$ chosen appropriately, the effect of the Tikhonov filter is similar to the truncation of the singular vector expansion performed by TSVD. The first singular components corresponding to the smaller indices are kept ($\phi_{\lambda,i} \approx 1$ for $i$ "small"), and the latter singular components corresponding to the larger indices are filtered away ($\phi_{\lambda,i} \approx 0$ for $i$ "large").

In the continuous case, we defined $\mathcal{R}_\alpha$ as a regularized operator in Definition 2.3. Similarly, in the discrete case, we denote the *regularized inverse* of the matrix $A$ by $A_\alpha^\#$. Thus for both TSVD and Tikhonov regularization the regularized inverse can be given as $A_\alpha^\# = V\Phi_\alpha\Sigma^\dagger U^T$, and by choosing the regularization parameter such that $\Phi_\alpha = I$ (the identity matrix), then $A_\alpha^\# = A^\dagger$ and we obtain the standard minimum-norm least squares solution.

### 2.2.3 General-Form Regularization

The smoothing-norm in the Tikhonov formulation can be more generally exchanged with some (semi-)norm described by the matrix $L$ such that the regularized solution is given by

$$x_{L,\lambda} = \operatorname{argmin}_x \left\{ \|b - Ax\|_2^2 + \lambda^2 \|Lx\|_2^2 \right\}. \tag{2.15}$$

If $L = I$ as in the previous sections, then the problem is said to be on *standard form*, otherwise the problem is said to be on *general form*. Often $L$ is chosen as some discrete approximation to a derivative operator such that the "flatness" or "smoothness" of the solution is restricted instead of the size of the solution. To analyze these kinds of problems, we start by introducing the generalized singular value decomposition.

**Definition 2.6** (GSVD) Let $A \in \mathbb{R}^{m\times n}$ and $L \in \mathbb{R}^{p\times n}$ be such that $m \geq n$ and $\mathcal{N}(A) \cap \mathcal{N}(L) = \{0\}$. Then the generalized singular value decomposition of the matrix pair $(A,\, L)$ is defined by

$$A = (\, U_1 \,,\, U_2 \,) \begin{pmatrix} \Sigma & \\ & \widehat{I} \end{pmatrix} \Theta^{-1}, \qquad L = V \,(\, M \,,\, 0 \,)\, \Theta^{-1},$$

where $(\, U_1 \,,\, U_2 \,) \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{p\times p}$ have orthonormal columns, $\Sigma \in \mathbb{R}^{p\times p} = \operatorname{diag}(\sigma_1, \ldots, \sigma_p)$ and $M \in \mathbb{R}^{p\times p} = \operatorname{diag}(\mu_1, \ldots, \mu_p)$ are diagonal, $\widehat{I} \in \mathbb{R}^{(m-p)\times(n-p)}$ is a matrix with ones on the main diagonal and zeros elsewhere, and $\Theta \in \mathbb{R}^{n\times n}$ is nonsingular. Furthermore, $\Sigma$ and $M$ are scaled such that $\Sigma^T\Sigma + M^T M = I$, and the generalized singular values are defined as $\gamma_i = \sigma_i/\mu_i$, for $i = 1, \ldots, p$ where $\gamma_1 \leq \cdots \leq \gamma_p$, such that $\Gamma = \operatorname{diag}(\gamma_1, \ldots, \gamma_p)$.

The solution can in this case be written as a linear combination of the columns of $\Theta = (\, \theta_1, \theta_2, \ldots, \theta_n \,)$, and the truncated GSVD (TGSVD) solution due to the regularization matrix $L$ can be written as

$$x_{L,k} = \sum_{i=p-k+1}^{p} \frac{u_i^T b^\delta}{\sigma_i}\theta_i + \sum_{i=p+1}^{n} (u_i^T b^\delta)\theta_i, \tag{2.16}$$

where the last term is the part of the solution in the possible nullspace of $L$, $\mathcal{N}(L)$. Similarly, the general-form Tikhonov solution can also be written in terms of the

GSVD. General-form regularization will be discussed in more detail in Chapter 5 where also two-dimensional problems are considered.

By generalizing the Tikhonov problem even further, we can exchange the 2-norms of the residual and the regularization term with other measures; e.g., we can formulate the general Tikhonov problem $\min_x \left\{ \|b^\delta - Ax\|_p^p + \lambda^q \|Lx\|_q^q \right\}$ where the residual and the solution are measured in two, possibly different, norms. If the noise in the right-hand side $b^\delta$ is normally distributed, then the least squares fit is a good choice. On the other hand, if there are outliers among the measurements, then the least squares fit will be severely affected by those, and choosing a more robust norm, e.g., the 1-norm $\|b^\delta - Ax\|_1$, will give a more robust estimate. For the solution norm, choosing a norm closer to one also makes the norm $\|Lx\|_q$ less sensitive to "outliers." That is, if $L$ is some approximation to a first derivative operator then the overall solution must be "flat," but for a norm closer to one, some jumps in the derivative are allowed which gives a possibility for more sharp changes in the solution.

Choosing norms other than 2-norms makes the problem considerably harder to solve because we cannot directly apply least squares algorithms. Several solution techniques are available for these kinds of problems, see, e.g., [76] for an approach using iteratively re-weighted least squares and several different norms, and [93] for an example of regularization using a total-variation approach.

## 2.3 Large-Scale Regularization

Direct computation of the SVD or GSVD are in general cumbersome and in practice impossible to do for large-scale problems. Therefore, explicitly calculating the Tikhonov or TSVD/TGSVD solutions in terms of filter factors is often impossible. Also, the system matrix $A$ may not be given explicitly, but only as a black-box function that implements the matrix-vector products $Ax$ for $x \in \mathbb{R}^n$ and possibly $A^T y$ for $y \in \mathbb{R}^m$. In this case it is impossible to compute any decomposition of $A$, and therefore iterative solution methods are often the only viable way.

Sometimes, the matrices are structured, e.g., circulant, such that clever decompositions exist. In these cases one might also for large-scale problems be able to compute directly filtered solutions in terms of either the singular values or the eigenvalues as we shall see later. Also, the iterative methods might exploit the structure to perform fast matrix-vector multiplications, e.g., using FFT-schemes.

Using iterative algorithms, we consider the same two basic ideas as described earlier, penalty methods and projection methods. That is, either we formulates a constrained optimization problem and use an iterative algorithm or a more general optimization scheme to solve this as good as possible, or we use the iterative methods to generate some solution subspace $\mathcal{S}_k$, which will hopefully be a suitable subspace for the regularized solution.

Hybrid methods are a slight extension to the projection methods where the projected problem is further regularized by applying direct or iterative regularization.

### 2.3.1 Penalty Approach

This approach is based on the Tikhonov problem and formulate the optimization problem by enforcing regularization explicitly though a penalty term such as $\|Lx\|_q^q$. The resulting minimization problem $\min_x\{\|b^\delta - Ax\|_p^p + \lambda^q\|Lx\|_q^q\}$ can then be solved by any suitable optimization scheme. In the general case, where the resulting optimization problem is possibly non-linear, any Newton-like or quasi-Newton-like method might be applied. See, e.g., [76] and the M$\mathcal{OO}$Re Tools algorithm gmin for such Newton type methods. A difficulty with this approach is that the $\lambda$ value must be chosen beforehand, and this choice is often not easy.

In a simpler case where $p = q = 2$, the resulting optimization problem – similar to the left equation in (2.11) – can be solved by any least squares solver. The more powerful methods belong to the class of Krylov subspace methods, and one can apply e.g., LSQR. Again, the $\lambda$ value must be chosen beforehand, and if several values must be tried then the entire problem must be solved several times. Moreover, the convergence of a least squares solver applied to the large least squares problem might be slow, which calls for efficient preconditioners. Unfortunately, it is not an easy task to identify efficient preconditioners for ill-conditioned problems.

### 2.3.2 Projection Approach

The second approach projects the problem onto a smaller subspace that hopefully provides a good basis for the solution. The TSVD method was earlier seen to project the solution onto the subspace span $\{v_1, v_2, \ldots, v_k\}$ where the $v_i$s are the right singular vectors of the system matrix $A$. It is well-known that at least certain iterative methods tend to generate subspaces that in a similar fashion are well suited for computing regularized solutions.

This regularizing effect has in particular been observed for least squares methods such as CGLS and LSQR [22, 33, 37, 73]. But for minimum-residual methods such as MINRES [78], GMRES [87], and variants of those, the situation is somewhat more "blurred." Some very promising attempts have been done to characterize the regularizing effect of the symmetric variants such as MINRES [31, 56], but for systems with general nonsymmetric coefficient matrices, GMRES has only been studied slightly [11]. One of the main aims of the following work is to investigate and analyze the regularization properties of MINRES and GMRES, as well as the variants MR-II [31] and RRGMRES [7].

### 2.3.3 Hybrid Approach

In many cases, the regularization obtained from projecting the solution onto a suitable solution subspace is sufficient to obtain a good regularized solution. But in some cases it might be relevant to consider an extra level of regularization and regularize also the projected problem. Moreover, it might be difficult to determine when to stop an iterative projection method, and inner regularization might be a help in

this connection [33, 59]. We will slightly discuss hybrid methods in connection with minimum-residual methods in Chapter 4.

## 2.4 The Regularization Parameter

From a theoretical point of view, Definition 2.3 implies that for any method to be a regularization method, one must be able to devise a parameter selection method such that the requirements of the definition are fulfilled. To do that, we need explicit knowledge of the noise level. This information is often not available, and we want to solve the problems approximately anyway.

Let us for a moment assume that we know the true solution $x_{\text{exact}}$. Then we can define the relative error of a computed solution compared to the true solution as

$$\varepsilon_p(x_\alpha) = \frac{\|x_{\text{exact}} - x_\alpha\|_p}{\|x_{\text{exact}}\|_p} \ . \tag{2.17}$$

Then we probably want to find the regularization parameter that minimizes (2.17). Often we choose $p = 2$ in (2.17), but other measures of the size may be suitable for some applications. When assessing the quality of a degraded image it is known that the 2-norm is not optimal and does not correlate well with the human perception. Attempts to incorporate a model of the Human Visual System (HSV) can provide a more suitable quality measure [75].

In a practical situation, however, the true solution is unknown, and we need to select a suitable parameter based on the sparse information we have available. Generally, we have only the discretized model $A$, the sampled right-hand side $b^\delta$, and maybe some knowledge of certain properties of the sought solution. In some cases, knowledge about the level of the noise in the measured right-hand side might be at hand; and this splits the parameter selection methods into two general classes — noise level based methods, and the methods that do not assume anything about the noise level.

### Discrepancy Principle

The (Morozov) discrepancy principle [68] is one of the most widely used methods for identifying a suitable regularization parameter [22, 29, 37]. The method requires knowledge of the noise level, and the philosophy is that we can never hope that the residual (or discrepancy) can be smaller than the errors in the input. More formally, let the discrepancy principle for selecting the regularization parameter be defined as

**Definition 2.7** The Discrepancy Principle defines the parameter $\alpha$ for which

$$\|b^\delta - Ax_\alpha\|_2 = \delta,$$

where $\delta \geq \|e\|_2$ is an upper bound for the noise in the right-hand side.

For TSVD and Tikhonov regularization, we have the following expressions for the norm of the residuals

$$\|b^\delta - Ax_k\|_2^2 = \sum_{i=k+1}^{n} (u_i^T b^\delta)^2 \quad , \quad \|b^\delta - Ax_\lambda\|_2^2 = \sum_{i=1}^{n} \left( \frac{\lambda^2}{\sigma_i^2 + \lambda^2} u_i^T b \right)^2 , \quad (2.18)$$

and obviously, the residual norm is nonincreasing for $k \to n$ and $\lambda \to 0$, respectively. For TSVD, where the regularization parameter is discrete, the discrepancy principle in Definition 2.7 may not be satisfied exactly. Therefore, one might reformulate the definition to

$$k_{\text{opt}} = \text{argmin}_k\{k\} \quad \text{s.t.} \quad \|b^\delta - Ax_k\|_2 \leq \delta$$

when the regularization parameter is discrete.

## Generalized Cross-Validation

The Generalized Cross-Validation (GCV) is a statistical method for choosing the regularization parameter without knowledge of the noise level. It tries to minimize the predictive mean-square error $\|b - Ax_\alpha\|_2$ where $b$ is the exact noise-free right-hand side $b = Ax_{\text{exact}}$ and $x_\alpha$ is a regularized solution due to the regularization parameter $\alpha$. We never have available the true right-hand side, and the GCV methods therefore aims at minimizing the function

$$\mathcal{G}(\alpha) = \frac{\|b^\delta - Ax_\alpha\|_2^2}{\text{trace}(I - AA^\#)^2},$$

where $A^\#$ is a regularized inverse such that $x_\alpha = A^\# b^\delta$. To formulate the GCV function, we therefore need to work with the regularized inverse and the trace term in the denominator which can be done when direct methods are applied. For example, for TSVD with truncation parameter $k$, the trace term is particularly simple because $A^\#$ in this case is given as $\sum_{i=1}^{k} \sigma_i^{-1} v_i u_i^T$ such that $I - AA^\# = m - k$. But for large-scale problems and when using iterative regularization, the regularized inverse is not unique [37, §7.4]. Formulating the GCV function when applying iterative regularization is therefore not straightforward.

## The L-curve

Another parameter estimation method is the so-called L-curve criterion. This method tries to find the parameter that describe an optimal trade-off between the minimization of the residual, and some measure of the computed solutions. By plotting corresponding values of these quantities in a log-log plot, it is often observed that the curve has a distinctive L-shape, and that the optimal regularization parameter is connected to a point near the corner of the L-curve. In general, the L-curve is described by the points

$$(\log \|b^\delta - Ax_\lambda\|_2, \log \Omega(x_\alpha)),$$

where $\Omega(x_\alpha)$ is some measure of the size of the solution. It is required for the L-curve to be well-defined that the residual norms are nonincreasing, and the solution norms are nondecreasing. Indeed for TSVD and Tikhonov regularization, Eq. (2.18) shows that the residual norms are nonincreasing for increasing $k$ and decreasing $\lambda$, respectively. Furthermore,

$$\|x_k\|_2^2 = \sum_{i=1}^{k} \left( \frac{u_i^T b}{\sigma_i} \right)^2 \quad , \quad \|x_\lambda\|_2^2 = \sum_{i=1}^{n} \left( \frac{\sigma_i}{\sigma_i^2 + \lambda^2} u_i^T b \right)^2 , \tag{2.19}$$

such that for increasing $k$ and decreasing $\lambda$, the solution norms are nondecreasing. Therefore, the L-curve is well-defined for both TSVD and Tikhonov regularization.

When the regularization parameter is continuous, e.g., if $\alpha = \lambda$ is the Tikhonov regularization parameter, then the L-curve is continuous and the corner can, in principle, be found as the point with maximum curvature.

In case the regularization parameter is discrete, e.g., if $\alpha = k$ is the truncation parameter for TSVD, or as we will see later the iteration number when performing iterative regularization, a similar *discrete L-curve* can be formulated. A problem with this discrete curve is that we have no operational expression for its second derivative. Consequently, we cannot easily compute the point of maximum curvature.

The L-curve is a heuristic that depends on whether or not a good solution is actually connected to the corner point of the L-curve. For smaller problems and for problems with fast decaying singular values, this is probably the case. But for large-scale problems, one must be careful when using the L-curve criterion. These issues, as well as a new strategy for locating the corner of a discrete L-curve, are dealt with in Chapter 7 and in [41].

## Normalized Cumulative Periodogram

The standard parameter choice methods mentioned above are in one way or another based on the norm of the residual, either alone, in combination with the solution (semi-)norm, or in statistical measures like the GCV function. A new parameter choice strategy that uses a normalized cumulative periodogram (NCP) [42] goes beyond the use of the residual norm and exploits instead the entire residual vector. The basic assumption is that in the noisy right-hand side $b^\delta = b + e$, the signal component $b$ and the noise component $e$ behave spectrally different. Due to the smoothing in the kernel, the true right-hand side $b$ is dominated by low-frequency components, whereas the additive noise $e$ is not. The parameter choice method might be based on a Kolmogorov-Smirnoff test of the NCP of the residual, i.e., a test whether the residual vector resembles white noise or not. If it does, then all components dominated by the true right-hand side $b$ has probably been extracted, and the remaining components are dominated by the noise $e$.

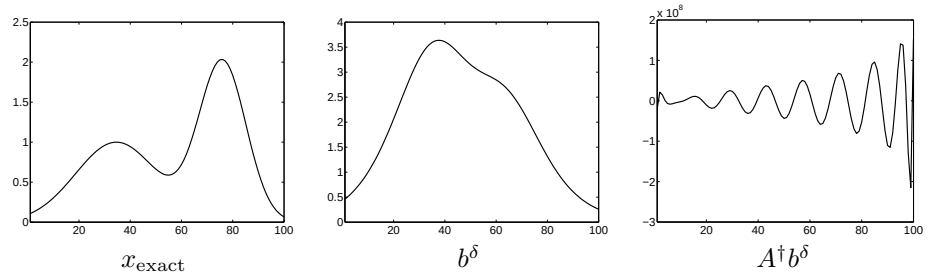



Figure 2.1: True solution $x_{\text{exact}}$, blurred and noisy right-hand side $b^\delta$, and the minimum-norm least squares solution $A^\dagger b^\delta$ to the shaw test problem. Note the axes of the right-most plot.

## 2.5 Numerical Example

To summarize this chapter, we illustrate the process of regularization with a simple example.

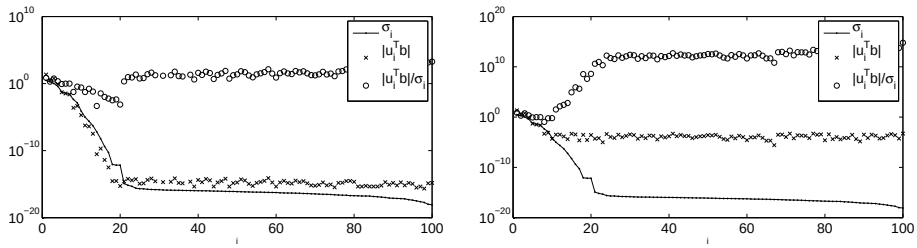
**Example 2.1** *Consider a one-dimensional image restoration problem where a blurred and noisy recording of a one-dimensional image should be restored. The continuous kernel and the true image are given as*

$$\begin{aligned} K(s,\,t) &= (\cos(s)+\cos(t))^2 \left(\frac{\sin(\pi(\sin(s)+\sin(t)))}{\pi(\sin(s)+\sin(t))}\right) \\ f(t) &= 2e^{-6(t-0.8)^2} + 2e^{-2(t+0.5)^2}, \end{aligned}$$

*and the integration intervals are $[-\pi/2\,;\,\pi/2]$ for both $s$ and $t$. The matrix $A$ and the discretized true solution $x_{\text{exact}}$ are generated with the function* shaw *from MOORe Tools [49] and obtained by simple collocation in $n = 100$ collocation points. The true right-hand side is constructed as $b = Ax_{\text{exact}}$, and the noise vector $e \in \mathbb{R}^n$ is constructed as white Gaussian noise with mean value 0 and scaled such that $\|e\|_2/\|b\|_2 = 10^{-3}$. The noisy right-hand side is then $b^\delta = b + e$.*

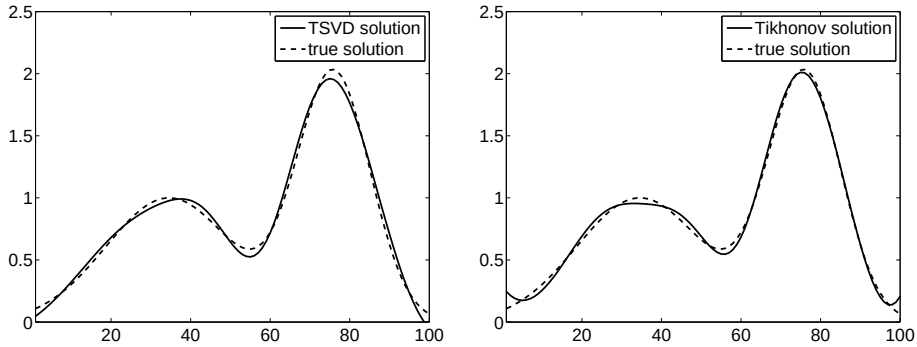
*Figure 2.1 shows the true solution $x_{\text{exact}}$, the blurred and noisy right-hand side $b^\delta$ as well as the naive solution $A^\dagger b^\delta$. Obviously, the naive solution does not resemble the true solution at all.*

*Next, we study the so-called Picard plot. Figure 2.2 shows two Picard plots; one for the true discrete right-hand side $b$, and one for the noisy right-hand side $b^\delta$. For $b$, we see that the components $|u_i^T b|$ fall off slightly faster than the singular values until both the right-hand side coefficients and the singular values hit the machine precision. This illustrates that even for the true right-hand side, the discrete Picard condition is most likely not satisfied due to the finite precision arithmetic and therefore numerical round-off errors. For the noisy right-hand side $b^\delta$, we see that the*




true right-hand side $b$ noisy right-hand side $b^\delta$

Figure 2.2: Picard plots for true discrete right-hand side $b$ and noisy discrete right-hand side $b^\delta$.



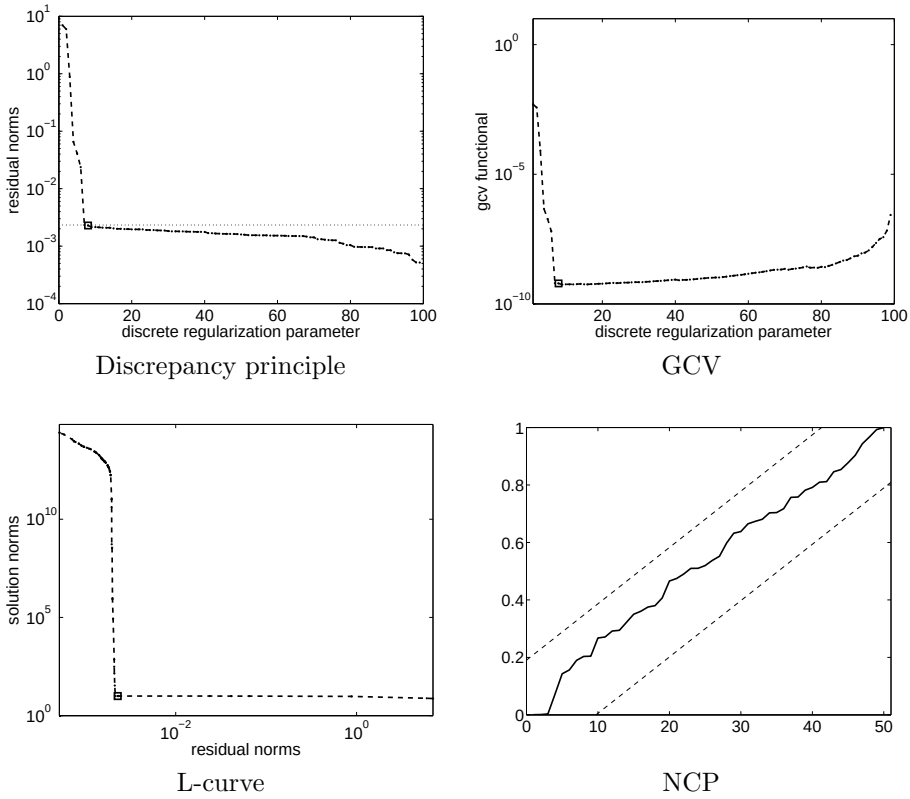

TSVD – $k = 8$ Tikhonov – $\lambda = 2.5 \cdot 10^{-4}$

Figure 2.3: TSVD and Tikhonov solutions to the shaw test problem.

*right-hand side coefficients $|u_i^T b^\delta|$ level off much earlier due to the additive noise. In fact, any singular component after index $i = 8$ will be mainly affected by the noise.*

*From the Picard plot it seems that a TSVD solution $x_k$ with truncation parameter $k = 8$ is a wise choice as a regularized solution. Indeed, the solution $x_8$ seen in the left plot of Fig. 2.3 approximates the true solution much better than the naive pseudoinverse solution. A similar Tikhonov solution, using the regularization parameter $\lambda = 2.5 \cdot 10^{-4}$, is seen in Fig. 2.3 (right).*

*Finally, we illustrate how the various parameter choice methods – discrepancy principle, GCV, the discrete L-curve, and the criterion based on the NCP – look for the TSVD solution. In Fig. 2.4 we see that the solution $x_8$ appears slightly after the residual has dropped below the level $\delta = \|b - b^\delta\|_2$, how $x_8$ corresponds to a point near the minimum of the GCV function, how $x_8$ appears near the corner of the L-curve, and finally how $x_8$ results in a residual $r_8 = b^\delta - Ax_8$ for which the NCP lies within the Kolmogorov-Smirnoff limits and therefore can be considered as white noise.*




Discrepancy principle GCV

L-curve NCP

Figure 2.4: Illustration of how the TSVD solution $x_8$ appears in connection with the parameter choice methods: discrepancy principle, GCV, the discrete L-curve, and the normalized cumulative periodogram (NCP). In the three first plots, the solution $x_8$ is indicated by □.

Chapter 3

# Image Deblurring

Deblurring of images is one application of regularization theory for large-scale problems. In fact, imaging in a broader sense covers not only reconstruction of everyday images, but also radar images, images arising from tomography, 3D images of the interior of the Earth, etc. Techniques and algorithms for dealing with this vast class of problems are important, and this chapter illustrates how image deblurring problems can be analyzed, and how different classes of matrix structures appear. Image deblurring is an active area of research and a lot of people have contributed in different ways [9, 17, 19, 34, 43, 69, 72, 74, 81, 88, 93]. The main contributions of this chapter is to formalize the results about blurring functions, study the matrix structures that appear when formulating two-dimensional image reconstruction problems, as well as to introduce some basic notation for the following chapters.

## 3.1 One-Dimensional Imaging

First, we consider deblurring in a simplified one-dimensional framework, i.e., we look at problems of the form (1.1) where $f(t)$ and $g(s)$ are one-dimensional signals or "images." The kernel $K(s,t)$ can describe different kinds of image distortions arising from imperfections in the focal system of a camera, atmospheric turbulence, motion of the camera, etc. All these distortions appear as some kind of blurring of the original scene $f(t)$ to the blurred realization $g(s)$. Therefore, $K(s,t)$ is often called a blurring kernel.

In turn, a blurring kernel is often described by a *point spread function* (PSF) which is a function that describes how one point (a pixel) in the true image is distorted in

the corresponding blurred image. There are two general properties of a PSF:

**Spatial Invariance.** The distortion of a pixel is independent of the location of the pixel. That is, the effect of the PSF is the same all over the image.

**Spatial Variance.** The distortion of a pixel changes with the location of the pixel. That is, a pixel is blurred differently depending on the location in the image.

In case the PSF is spatially invariant then the blurring process can be described as a convolution of the PSF and the image, i.e., the PSF only depends on the difference between the location of a pixel in the true and the blurred images [51, §2.3]. Furthermore, the PSF is often local, i.e., a pixel in the true image only affects nearby pixels in the blurred image. Consider now a discretization of a PSF

$$t = (\dots, 0, t_{-\beta+1}, \dots, t_{-1}, t_0, t_1, \dots, t_{\beta-1}, 0, \dots)^T. \tag{3.1}$$

Due to the locality of the PSF, the contributions decay with the distance to the center point of the PSF, and from a certain distance $\beta$ from the center, the contributions can be considered to be negligible. This distance $\beta$ is called the bandwidth. Similarly, a discrete realization $\tilde{x} \in \mathbb{R}^{n+2\beta}$ of the true image $f(t)$ is considered, i.e., $\tilde{x} = (\tilde{x}_{-\beta+2}, \dots, \tilde{x}_0, \tilde{x}_1, \dots, \tilde{x}_n, \tilde{x}_{n+1}, \dots, \tilde{x}_{n+\beta-1})^T$ where again $\beta$ is the bandwidth of the PSF. We can apply the discrete PSF to get all $n$ points in the discrete blurred image $b$, and we get by convolution

$$b_i = \sum_{j=-\beta+2}^{n+\beta-1} t_{i-j}\tilde{x}_i, \quad i = 1, 2, \dots, n.$$

Now we formulate a PSF matrix based on (3.1) that, when multiplied to the vector $\tilde{x}$, describes all the points of the discrete blurred image $b$

$$\widetilde{T} = \begin{pmatrix} t_{-\beta+1} & \dots & t_0 & \dots & t_{\beta-1} & 0 & \dots & \\ 0 & t_{-\beta+1} & \dots & t_0 & \dots & t_{\beta-1} & \ddots & \\ & \ddots & \ddots & & \ddots & & \ddots & \ddots \\ & & 0 & t_{-\beta+1} & \dots & t_0 & \dots & t_{\beta-1} \end{pmatrix} \in \mathbb{R}^{n\times(n+2\beta)}. \tag{3.2}$$

Because the PSF is spatially invariant then $\widetilde{T}$ has Toeplitz structure, and obviously the system of equations $\widetilde{T}\tilde{x} = b$ is underdetermined. To address this issue, we need to consider boundary conditions.

### 3.1.1 Boundary Conditions and Matrix Structures

There are a number of different boundary conditions (BCs) that can be incorporated into the coefficient matrix. In accordance with e.g., [74] and [88], we split up the discrete one-dimensional image and the full blurring matrix (3.2) into three parts such that

$$\overline{T}\overline{x} + Tx + \underline{T}\underline{x} = b, \tag{3.3}$$

where $\tilde{x} = (\overline{x}, x, \underline{x})^T$ with $x = (\tilde{x}_1, \ldots, \tilde{x}_n)^T$, and $\overline{x}$ and $\underline{x}$ have length $\beta$ and contain the parts of $\tilde{x}$ that "spill over" the boundaries. Now $T$ is the central $n \times n$ part of $\widetilde{T}$, and $\overline{T}$ and $\underline{T}$ have size $n \times \beta$. Furthermore, let $J \in \mathbb{R}^{n \times n}$ be the $n \times n$ reversal matrix, i.e., a matrix with ones on the antidiagonal. We now have several options for choosing the BC. In each case, we define the PSF matrix $A$ based on the full rectangular PSF matrix (3.2) such that it includes the desired BCs.

**Zero BC.** Assume that the signal outside the studied domain is zero. That is,

$$\tilde{x}_i = 0 \quad \text{for} \quad i = -\beta + 2, \ldots, 0, n + 1 \ldots, n + \beta - 1$$

Then $\overline{T}\overline{x} = 0$ and $\underline{T}\underline{x} = 0$, and thus the model of the PSF including zero boundary conditions is

$$A = T. \tag{3.4}$$

That is, from a matrix point of view, zero boundary conditions are simple and results in disregarding the parts of the signal that "spill over" the boundaries.

**Periodic BC.** Assume that the signal repeats itself outside the region. That is,

$$\tilde{x}_i = \tilde{x}_{n+i} \quad \text{for} \quad i = -\beta + 2, \ldots, \beta - 1$$

and the model of the PSF can be formulated as

$$A = \left[ (Z, \overline{T}) + T + (\underline{T}, Z) \right], \tag{3.5}$$

where $Z \in \mathbb{R}^{n \times (n-\beta)}$ are zero matrices. For periodic boundary conditions, the parts of $\widetilde{T}$ that "spill over" the boundaries are therefore copied back into the opposite side of the domain.

**Reflexive BC.** Assume that the signal is reflected across the boundaries, i.e.,

$$\begin{aligned} \tilde{x}_i &= \tilde{x}_{1-i} && \text{for} \quad i = -\beta + 2, \ldots, 0 \\ \tilde{x}_{n+i} &= \tilde{x}_{n-i+1} && \text{for} \quad i = 1, \ldots, \beta - 1 \end{aligned}$$

The model that incorporates the reflexive boundary conditions can be described by

$$A = \left[ (Z, \overline{T})\, J + T + (\underline{T}, Z)\, J \right], \tag{3.6}$$

where again $Z \in \mathbb{R}^{n \times (n-\beta)}$ are zero matrices. Here, the parts that "spill over" the boundaries are not copied to the opposite side of the domain, but reflected back into the domain.

**Antireflexive BC.** Recently, the antireflexive boundary conditions appeared in the literature [19, 88]. These are defined by

$$\begin{aligned} x_{1-j} &= 2x_1 - x_{j+1} \quad \text{for} \quad j = 1, \ldots, \beta - 1 \\ x_{n+j} &= 2x_n - x_{n-j} \quad \text{for} \quad j = 1, \ldots, \beta - 1. \end{aligned}$$

To formulate an expression for $A$ similar to those above, we have to consider an additional rank-one correction for each of the boundaries which complicates the expression. We refer to [88] for a more thorough analysis, and state here only that the blurring matrix $A$ can be formulated as

$$A = \left[ ze_1^T - (\, Z \,,\, \overline{T}\,)\, \widetilde{J} + T - (\, \underline{T} \,,\, Z\,)\, \widehat{J} + we_n^T \right], \tag{3.7}$$

where $e_1$, $e_n \in \mathbb{R}^n$ are the first and last canonical basis vectors of $\mathbb{R}^n$, the vectors $z$, $w \in \mathbb{R}^n$ are based on the blurring, and $\widetilde{J}$ and $\widehat{J}$ are given by

$$\widetilde{J} = \begin{pmatrix} 0 & \\ & J \end{pmatrix} \quad , \quad \widehat{J} = \begin{pmatrix} J & \\ & 0 \end{pmatrix}.$$

The different boundary conditions described above give rise to different structures of the coefficient matrix. These are described in the following.

### Periodic BC – Circulant Matrices

Circulant matrices arise when the periodic boundary conditions are used, and is a particularly favorable structure that has the following generic form

$$C = \begin{pmatrix} c_0 & c_{n-1} & \cdots & c_1 \\ c_1 & \ddots & & c_2 \\ \vdots & \ddots & & \vdots \\ c_{n-1} & c_{n-2} & & c_0 \end{pmatrix},$$

i.e., a matrix with constant values along all diagonals. The columns "wrap around" such that the matrix is completely determined by the first column or the first row, i.e., by $n$ elements. This fact decreases the storage requirements considerably from $n^2$ to $n$. Furthermore, all circulant matrices are diagonizable by the discrete Fourier transform such that $C = F\Lambda F^H$, where $F \in \mathbb{C}^{n\times n}$ is the orthogonal Fourier basis, $\Lambda \in \mathbb{C}^{n\times n}$ is a diagonal matrix containing the eigenvalues of $C$, and $F^H$ is the conjugate transpose of $F$. The matrix $F$ needs never be formed explicitly, as it is implemented through the fast Fourier transform (FFT) which has complexity $\mathcal{O}(n \log n)$. Applying a circulant matrix $C$ to a vector can therefore be done by performing two FFTs and $n$ complex products. The overall complexity is therefore $\mathcal{O}(n \log n)$ which should be compared to the $\mathcal{O}(n^2)$ complexity for a standard matrix-vector product.

### Zero BC – Toeplitz Matrices

Imposing zero boundary conditions results in a Toeplitz matrix which has the generic form

$$T = \begin{pmatrix} t_0 & t_{-1} & \dots & t_{-n+1} \\ t_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \\ t_{n-1} & \dots & & t_0 \end{pmatrix}.$$

Any Toeplitz matrix is completely defined by the first row and the first column, i.e., $2n-1$ elements. A Toeplitz matrix is not easily diagonizable by itself, but it can easily be embedded into a larger circulant matrix of size $(2n-1) \times (2n-1)$. Thus, a matrix-vector product can again benefit from the FFT and be performed in $\mathcal{O}(n \log n)$.

### Reflexive BC – Toeplitz-plus-Hankel Matrices

The reflexive boundary conditions result in a Toeplitz-plus-Hankel matrix [71, 74]. A Hankel matrix is similar to a Toeplitz matrix, but have constant values along the antidiagonals

$$H = \begin{pmatrix} h_{-n+1} & \dots & h_{-1} & h_0 \\ \vdots & \iddots & \iddots & h_1 \\ & \iddots & \iddots & \vdots \\ h_0 & & & h_{n-1} \end{pmatrix}.$$

Again, any Hankel matrix is completely determined by $2n-1$ elements. When constructing the blurring matrix from a PSF with reflexive boundary conditions, the same values are used to make up both the Toeplitz matrix $T$ and the Hankel matrix $H = (\, Z \,,\, \overline{T} \,)\, J + (\, \underline{T} \,,\, Z \,)$ in (3.6), so the overall storage requirements are the same. Furthermore, matrix-vector multiplications with a Hankel matrix can also be implemented in $O(n \log n)$. This is easily seen because $Hx = (HJ)(J^T x)$ where $J$ is the $n \times n$ reversal matrix, and $HJ$ therefore have Toeplitz structure. Matrix-vector multiplications with $A = T + H$ can therefore again be implemented in $O(n \log n)$ by multiplying with $T$ and $H$ separately.

A very important special case is when the PSF is symmetric, i.e., $t_{-i} = t_i$, $\forall\, i$. In this case the entire Toeplitz-plus-Hankel matrix $A = T + H$ is diagonalized by the discrete cosine transform type II (DCT-II) [45, 51, 74, 82]. Particularly, Lemma 3.1 and Theorem 3.2 from [74] connect the structure of the PSF matrices with the class of matrices that are diagonalized by the DCT. This approach results in a more efficient $O(n \log n)$ algorithm for performing matrix-vector multiplication because only one transformation is needed, and because the DCT is a real transform, whereas the FFT is complex. Furthermore, the DCT diagonalizes the entire $A$ which can be used to efficiently compute Tikhonov solutions.

For future reference, we define the DCT transform matrix.

**Definition 3.1** (DCT-II) Let the one-dimensional discrete cosine transform matrix $\Psi \in \mathbb{R}^{n\times n}$ have the elements

$$\psi_{i,j} = w_j \cos\left(\frac{(i-1)(2j-1)\pi}{2n}\right), \quad 1 \le i,\, j \le n,$$

where $w_j = \sqrt{1/n}$ for $j = 1$ and $w_j = \sqrt{2/n}$ for $j > 1$.

### Antireflexive BC – Toeplitz-plus-Hankel-plus-Rank-2

The less well-known antireflexive boundary conditions lead to an even more exotic matrix structure. As described in e.g., [88, 81] and seen in (3.7), the resulting blurring matrix can be described by the direct sum of a Toeplitz matrix, a Hankel matrix, and a rank-2 correction. This structure is again memory efficient in its storage requirements.

Again, an important special case arises when the PSF is symmetric. In this case the PSF matrix is, when the rank-2 correction is subtracted, diagonizable by the discrete sine transform type-I (DST-I) [82].

### Symmetric Blurring Matrix

We saw above that the symmetry of the PSF is an important property when implementing reflexive and antireflexive boundary conditions because clever diagonalization schemes can then be used both to factorize the matrices and to implement fast matrix-vector multiplications. But as we shall see later in connection with the study of Krylov subspace methods, the symmetry of the PSF matrix $A$ is also important. Unfortunately, the symmetry of the PSF matrix depends not only on the PSF itself, but also on the boundary conditions. Moreover, a symmetric PSF matrix may describe a nonsymmetric and spatially variant PSF.

**Theorem 3.2** *If the one-dimensional PSF* (3.1) *is symmetric and spatially invariant then zero, periodic, and reflexive boundary conditions lead to symmetric PSF matrices, but antireflexive boundary conditions do not.*

Proof. When the PSF is spatially invariant, then the full PSF matrix from (3.2) has Toeplitz structure. Consider again the splitting from (3.3). If the PSF is symmetric, then the central part of the full blurring matrix $T$ is symmetric, and the "spill over" in both sides of the domain are identical. Therefore, $A$ implementing zero boundary conditions (3.4) is symmetric, $A$ implementing periodic boundary conditions (3.5) is symmetric because $(\, Z\ \overline{T}\,) = (\, \underline{T}\ Z\,)^T$, and $A$ implementing reflexive boundary conditions (3.6) is symmetric because any Hankel matrix is symmetric, i.e., $(0\ \overline{T})J + (\underline{T}\ 0)J$ is symmetric. Antireflexive boundary conditions obviously lead to a nonsymmetric $A$ because the rank-2 correction $ze_1^T + we_n^T$ is not in general symmetric. □

It is straightforward to see that in very special cases, a spatially variant PSF might result in a symmetric PSF matrix. While this will probably seldom happen for real deblurring problems, it does show that the symmetry of the PSF matrix does not necessarily imply symmetry and spatial invariance of the PSF.

**Example 3.1** *Consider the following symmetric PSF matrix that implements zero BCs*

$$A = \begin{pmatrix} t_{0,0} & t_{0,1} & t_{0,1} & \dots & t_{0,n-1} \\ t_{0,1} & t_{1,0} & t_{1,1} & \ddots & \vdots \\ t_{0,2} & t_{1,1} & t_{2,0} & \ddots & \\ \vdots & \ddots & \ddots & \ddots & \\ t_{0,n-1} & \dots & & & t_{1,n-1} \end{pmatrix}.$$

*Each row describes a different PSF (spatial variance), but the rows are such that the resulting PSF matrix A is symmetric.*

### 3.1.2 Summary

In summary, we note that any spatially invariant PSF (symmetric or not) will result in a blurring matrix with a structure depending on the imposed BCs. In the case we apply periodic boundary conditions, then the FFT can be applied directly to the resulting circulant PSF matrix which leads to fast algorithms for computing matrix-vector products, as well as diagonalizing the matrix. For the remaining boundary conditions, fast matrix-vector multiplications can be performed by embedding the resulting matrices into a larger circulant matrix and then an FFT approach can be used. A direct diagonalization of a matrix that implements zero boundary conditions is not in general easy, and for reflexive and antireflexive BCs a diagonalization is only easy if the PSF is symmetric. If the PSF is spatially invariant and symmetric, then the PSF matrix is also symmetric, except if it implements antireflexive BCs. Spatially variant PSFs do not result in any of the above structured matrices, but might in special cases lead to a symmetric blurring matrix.

For all of the above mentioned structured matrices, we refer to [81] for a nice description of all of them in a common framework.

## 3.2 2D Image Deblurring Problems

Now we focus on two-dimensional problems, and in the continuous domain images can be considered as real-valued two-dimensional functions $f(s,t)$. In a more general setting, also complex-valued functions can be considered, e.g., if the image results from electromagnetic prospering. The inverse problem is now based on the two-

dimensional Fredholm integral equation of the first kind

$$\int_0^1 \int_0^1 K(s, \bar{s}, t, \bar{t}) f(s, t)\, ds\, dt = g(\bar{s}, \bar{t}), \quad 0 \leq \bar{s},\, \bar{t} \leq 1, \tag{3.8}$$

where $K(s, \bar{s}, t, \bar{t})$ is the kernel, and $g(\bar{s}, \bar{t})$ is the exact blurred version of the true image $f(s, t)$. In a more general setting, the number of variables in $f$ and $g$ may be different, e.g., if the image of an original three-dimensional domain is measured on a two-dimensional surface. This situation can arise e.g., in geomagnetic prospering where the sought solution is a function in three variables $f(x, y, z)$ under the surface of the Earth, and the measurements are carried out in a two-dimensional plane over the surface, i.e., $g(\bar{x}, \bar{y})$ is a function in only two variables.

Image deblurring problems are often large-scale, and it is important to exploit any possible structure – either such that decomposing the PSF matrix becomes feasible, or such that applying the PSF matrix to a vector can be done efficiently.

### 3.2.1 Discretized Images

As for one-dimensional problems, we again discretize the problems to obtain a problem in the general framework of systems of linear equations, and we end up with "two-dimensional vectors" that represent the images. To put these into the common framework, we define the following functions.

**Definition 3.3** (vec) The vec function transforms the image $X \in \mathbb{R}^{m \times n}$ into a vector of length $mn$

$$\mathrm{vec}\,(X) = \left( x_1^T\,,\, x_2^T\,,\, \ldots\,,\, x_m^T \right)^T,$$

i.e., $\mathrm{vec}\,(X)$ is a column-wise stacking of the columns of $X$. The definition easily generalizes to problems of higher dimensions.

**Definition 3.4** ($\mathrm{vec}^{-1}$) The $\mathrm{vec}^{-1}$ function performs the inverse operation of the vec function. Additional information of the size of the image is needed. Let $x \in \mathbb{R}^{mn}$ be a vector representing an image of dimensions $m \times n$, then

$$\mathrm{vec}^{-1}\,(x, m, n) = X,$$

where $X \in \mathbb{R}^{m \times n}$ is the original image.

The blurring kernels for image problems should be discretized according to the discretization and stacking of the images, and the resulting linear systems are often very large.

### 3.2.2 Blur

For two-dimensional images there are several different sources for blurring; e.g, out-of-focus blur, in which case the light intensity that should appear for one single

element of the image is smeared out and averaged over a number of connected pixels in both directions. Motion blur can be described in a similar manner where the out-of-focus effect appears in a specific direction according to the motion. A third effect that is often used for illustrative purposes (and indeed also in this thesis) is atmospheric turbulence blur [38] which can often be described as a two-dimensional Gaussian

$$K(s, \bar{s}, t, \bar{t}) = \frac{1}{2\pi\sigma_s\sigma_t} \exp\left(-\frac{1}{2}\left(\frac{s-\bar{s}}{\sigma_s}\right)^2 - \frac{1}{2}\left(\frac{t-\bar{t}}{\sigma_t}\right)^2\right), \tag{3.9}$$

where $\sigma_s$ and $\sigma_t$ denote the width of the blurring in the two axis directions. Other models for such blurrings exist, see, e.g., Moffat [67].

The types of blurring mentioned above all describe spatially invariant blur. In many real-world applications this is not the case, and one has to formulate more complicated models for the spatially variant blur. The main focus of the present thesis is not on particular image deblurring problems, and we will therefore not go into detail with more specific blurring models. Some authors investigate models of spatially variant blurs that are constructed, e.g., by interpolating the effect of several spatially invariant blurring operators [70, 71]. See also [43] for a general reference to image deblurring problems.

### 3.2.3 Matrix Structures

Again, the function that describes the blurring of an image is referred to as a PSF, and in the two-dimensional case there are some additional properties to consider. Assume that the bandwidth $\beta$ of the PSF is the same in the two axis directions, and define the two-dimensional discrete PSF as the PSF image

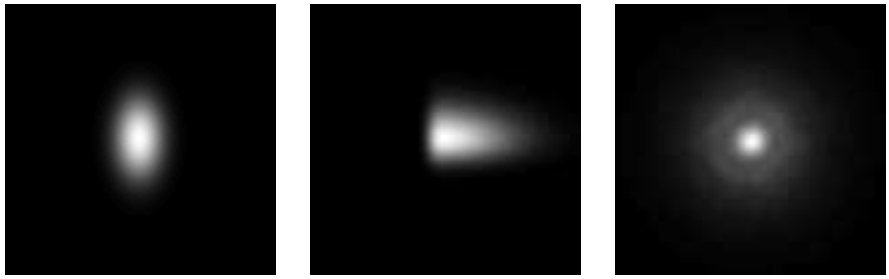
$$H = \begin{bmatrix} h_{-\beta,-\beta} & \dots & h_{-\beta,0} & \dots & h_{-\beta,\beta} \\ \vdots & & \vdots & & \vdots \\ h_{0,-\beta} & \dots & h_{0,0} & \dots & h_{0,\beta} \\ \vdots & & \vdots & & \vdots \\ h_{\beta,-\beta} & \dots & h_{\beta,0} & \dots & h_{\beta,\beta} \end{bmatrix}. \tag{3.10}$$

Furthermore, assume that elements outside the bandwidth are zero, i.e., $h_{i,j} = 0$ for all $|i|, |j| > \beta$. Apart from spatial variance and invariance, the additional properties are

**Separable.** The PSF separates such that the the blurring can be applied separately to the rows and columns of the image. Define two PSF matrices $A_1 \in \mathbb{R}^{m\times m}$ and $A_2 \in \mathbb{R}^{n\times n}$, then the blurring can be written in matrix form as

$$B = A_1 X A_2^T, \tag{3.11}$$

where $X \in \mathbb{R}^{m\times n}$ is the true image. Moreover, for a separable PSF, the PSF image $H$ from (3.10) is a rank-one matrix, see, e.g., [54, 69].



(a) (b) (c)

Figure 3.1: Examples of PSFs. The point function is located in the center of the images. (a) Symmetric and separable, (b) nonsymmetric, but separable, and (c) nonsymmetric and nonseparable.

**Symmetric.** For a two-dimensional PSF to be symmetric, it needs to be symmetric along both image axes. That is, $h_{-i,j} = h_{i,j} = h_{i,-j} = h_{i,j}$ for all $i$ and $j$.

Again, a common property is that the blurring is local, which means, a pixel $X_{i,j}$ affects only a limited amount of connected pixels in the blurred image $B$. Examples of PSFs are seen in Fig. 3.1. The first PSF is separable and symmetric, the second is still separable but nonsymmetric in both axis directions, and the last PSF is neither symmetric, nor separable.

When discretizing a PSF for applying it to a stacked image $\mathrm{vec}\,(X)$ where $X \in \mathbb{R}^{m\times n}$ to get the blurred image $\mathrm{vec}\,(B)$ where $B \in \mathbb{R}^{m\times n}$, we get a PSF matrix $A \in \mathbb{R}^{mn\times mn}$. Even for smaller images $X$ and $B$, the resulting PSF matrix is huge. It is therefore of utmost importance to exploit any possible structure of these matrices. The matrix structures described for the one-dimensional case generalizes quite naturally to higher dimensions and the PSF matrices become block matrices.

**Periodic BCs.** This choice of BCs results in matrices that are block circulant with circulant blocks (BCCB). It is well-known that BCCB matrices are diagonalized by the 2D-FFT (see e.g., [51]). The cost of a matrix-vector product is therefore only $O(n^2 \log n)$ compared to $O(n^4)$. For this reason, periodic boundary conditions are often used in signal and image processing.

**Zero BCs.** The Toeplitz structure from the one-dimensional case generalizes to matrices that are block Toeplitz with Toeplitz blocks (BTTB). These can be embedded into larger BCCB matrices and again the 2D-FFT can be used to obtain $O(n^2 \log n)$ algorithms for matrix-vector multiplications, but not for diagonalization of the PSF matrix.

**Reflexive BC.** Reflexive boundary conditions result in matrices that are block Toeplitz-plus-Hankel with Toeplitz-plus-Hankel blocks (BTHTHB). The struc-

ture can also be split up into a sum of four components, namely a BTTB, BTHB, BHTB, and BHHB [69].

A result by Ng et al. [74, Theorem 3.3] states that when the PSF is symmetric, i.e., when $h_{-i,j} = h_{i,j} = h_{i,-j} = h_{i,j}$ for all $i$ and $j$, then a PSF matrix $A$ that implement reflexive boundary conditions can be diagonalized by the 2D-DCT. The two-dimensional DCT matrix can be defined as $\Psi \otimes \Psi$ where $\Psi \in \mathbb{R}^{n\times n}$ is defined in Definition 3.1, and thus $(\Psi \otimes \Psi)A(\Psi \otimes \Psi)^T = \Lambda$ is diagonal.

**Antireflexive BCs.** These boundary conditions also lead to a block structure of the PSF matrices that can now be described as block Toeplitz-plus-Hankel-plus-Rank-2 with Toeplitz-plus-Hankel-plus-Rank-2 blocks (BTHR2THR2B). Unfortunately, there are two possible extensions to the image near the corners, and the resulting PSF matrix is therefore not unique. The two extensions are 1) antireflection around the corners, and 2) antireflection along the two axes independently (see [88] and especially [18] for discussions and illustrations). In case the PSF is symmetric, it is still possible to devise $O(n^2 \log n)$ algorithms for matrix-vector multiplication based on the 2D-DST.

Thus in several cases, efficient matrix-vector products can be implemented also for the much larger two-dimensional block matrices.

### Separability

A very special block structure arise for separable PSFs. We start by defining the Kronecker product in the following way

**Definition 3.5** (Kronecker product) Let $A \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{p\times q}$ be two matrices and let the elements of $A$ be $a_{ij}$ for $i = 1, \ldots, m$, and $j = 1, \ldots, n$. Then the Kronecker product $C = A \otimes B \in \mathbb{C}^{mp\times nq}$ is defined by

$$C = A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \ldots & a_{1n}B \\ \vdots & \ddots & & \vdots \\ a_{m1}B & \ldots & & a_{mn}B \end{pmatrix}.$$

Furthermore, the following Lemma describes several important properties of the Kronecker product.

**Lemma 3.6** *Several properties hold for the Kronecker Product*

1. $A_1 X A_2^T = B \Leftrightarrow (A_2 \otimes A_1)\mathrm{vec}\,(X) = \mathrm{vec}\,(B)$.
2. $(A_1B_1 \otimes A_2B_2) = (A_1 \otimes A_2)(B_1 \otimes B_2)$
3. $(A_1 \otimes A_2)^T = A_1^T \otimes A_2^T$
4. $(A_1 \otimes A_2)^{-1} = A_1^{-1} \otimes A_2^{-1}$

It is easily observed that using the first relation from Lemma 3.6, the system (3.11) based on the two blurring matrices $A_1$ and $A_2$ can be written in standard form as

$$(A_2 \otimes A_1)\text{vec}\,(X) = \text{vec}\,(B) \;\Rightarrow\; Ax = b,$$

where $A \in \mathbb{R}^{mn\times mn}$, and $x$, $b \in \mathbb{R}^{mn}$.

Assume that the image $X$ is square, i.e., $m = n$. We see that the memory requirements for storing the full blurring matrix $A \in \mathbb{R}^{n^2\times n^2}$ reduces from $n^4$ to $2n^2$ elements if the separable blurring is stored as $A_1$ and $A_2$. Furthermore, a matrix-vector product $Ax$ costs $2n^4 - n^2$, whereas performing the equivalent matrix-matrix multiplications $A_2XA_1$ costs in total $4n^3 - 2n^2$ operations. In this comparison, the possible structure of $A_1$ and $A_2$ are not considered. If these implement symmetric and spatially invariant PSFs with some boundary conditions, then the structures from Section 3.1.1 for one-dimensional problems hold, and the fast one-dimensional transforms can be used for $A_1$ and $A_2$ individually to implement $O(n^2 \log n)$ algorithms for $Y = A_2X$ and $YA_1$. This obviously holds also when $A_1$ and $A_2$ implement different boundary conditions, whereas the block structures described earlier arise only if the same boundary conditions are used in both directions.

We note that the separability property of the PSF is independent of whether the PSF is spatially variant or invariant, i.e., a PSF can be spatially variant and still separable. In this case $A$ will have no particular block structure, but still be the result of a Kronecker product.

### 3.2.4 Approximate Blurring Matrices

If the PSF is not symmetric then the blurring matrices arising from imposing reflexive or antireflexive boundary conditions are no longer diagonizable by the 2D-DCT and the 2D-DST. If the PSF is spatially variant, then the blurring matrix will not possess any of the convenient matrix structures mentioned in Section 3.1.1.

If the PSF is still strictly separable then it can be formulated as a Kronecker product of two smaller matrices as described above. But in the general case one might need to formulate an approximation of the blurring matrix; both because of storage requirements and to do efficient computations with the large blurring matrix.

A lot of work has been done trying to approximate general spatially invariant PSFs with sums of Kronecker products, see e.g., [54, 69, 81], as well as several implementations in the image reconstruction toolbox Restoration Tools by Nagy, Palmer, and Perrone [71]. A general idea is to find matrices $A_k$ and $B_k$ for $k = 1, \ldots, s$ such that

$$\min \left\| K - \sum_{k=1}^{s} (A_k \otimes B_k) \right\|_F ,$$

where $K$ is the large, sparse, and possibly unstructured, PSF matrix, and $A_k$ and $B_k$ are structured matrices corresponding to some chosen boundary conditions. The parameter $s$ determines the number of terms in the sum, i.e., $s = 1$ for a perfectly separable blurring.

It is observed [81] that often a fairly low number of terms are able to approximate the blurring matrix, even in the case where the PSF is non-separable.

### 3.2.5 Overview

The properties of the PSF and the corresponding matrix structures are combined in Table 3.1. All spatially invariant PSFs give rise to structured PSF matrices of different kinds depending on the boundary conditions. Independent of this, the matrix is the result of a Kronecker product of two smaller matrices whenever the blurring separates. The diagonizability of the coefficient matrix through 2D-FFT, 2D-DCT, or 2D-DST does not depend on the separability but solely on the boundary conditions. Moreover, the symmetry of the PSF matrix for spatially invariant PSFs depend not only on the symmetry of the PSF, but also on the imposed boundary conditions.

Spatially variant PSFs in general have no circulant, Toeplitz, or Hankel structure, but might still be a result of a Kronecker product of two smaller matrices. Furthermore, the resulting PSF matrix may in special cases be symmetric even though the blurring of each pixel is not symmetric.

| PSF | Boundary | PSF Matrix | Diagonizable | Kronecker |
|---|---|---|---|---|
| inv/sep/sym | periodic | BCCB/sym | 2D-FFT | yes |
| — | zero | BTTB/sym | no | yes |
| — | reflexive | BTHTHB/sym | 2D-DCT | yes |
| — | antireflexive | BTHR2THR2B/nonsym | r2 + 2D-DST | yes |
| inv/sep/nonsym | periodic | BCCB/nonsym | 2D-FFT | yes |
| — | zero | BTTB/nonsym | no | yes |
| — | reflexive | BTHTHB/nonsym | no | yes |
| — | antireflexive | BTHR2THR2B/nonsym | no | yes |
| inv/nonsep/sym | periodic | BCCB/sym | 2D-FFT | no |
| — | zero | BTTB/sym | no | no |
| — | reflexive | BTHTHB/sym | 2D-DCT | no |
| — | antireflexive | BTHR2THR2B/nonsym | r2 + 2D-DST | no |
| inv/nonsep/nonsym | periodic | BCCB/nonsym | 2D-FFT | no |
| — | zero | BTTB/nonsym | no | no |
| — | reflexive | BTHTHB/nonsym | no | no |
| — | antireflexive | BTHR2THR2B/nonsym | no | no |
| variant/sep | any | no but maybe sym | no | yes |
| variant/nonsep | any | no but maybe sym | no | no |

Table 3.1: Overview of properties for point spread functions (PSFs) and the corresponding matrix structures. The PSFs can be invariant/variant, separable/nonseparable, and symmetric/nonsymmetric. The description of the block structure of the PSF matrix follows the notation in the previous chapters.

## 3.3 Regularization of Image Deblurring Problems

The blurring process often leads to an ill-posed problem, and the solution needs to be stabilized. So far we saw an example using TSVD and Tikhonov regularization to stabilize the solution of the one-dimensional image reconstruction problem in Example 2.1. The basis for both of these regularization methods is that the discrete Picard condition is satisfied, at least for the initial SVD components until the noise-level or the effects of the finite precision arithmetic disturb the components. For one-dimensional kernels for ill-posed problems it was justified in e.g., [37] that the singular vectors has a smoothing property such that a low index corresponds to a low-frequent singular vector, whereas a high index corresponds to a high-frequent singular vector. A similar observation was done for two-dimensional blurring matrices in [52], i.e., TSVD and Tikhonov regularization will have a similar effect for two-dimensional problems.

If we again want to analyze the problems through Picard plots and directly compute TSVD and Tikhonov solutions, then we need the SVD of the large PSF matrix $A$. In general, if $A$ is large and unstructured, then computing the SVD is challenging, to put it mildly. But if we assume that $A$ is separable then the Kronecker product provides a way to efficiently compute the SVD of the blurring matrix, and direct computation of TSVD or Tikhonov solutions are possible.

**Lemma 3.7** *(Kronecker SVD) Given the Kronecker product $K = A \otimes B$, and the SVDs of the small matrices $A = U_A \Sigma_A V_A^T$ and $B = U_B \Sigma_B V_B^T$, the SVD of $K$ is given as*

$$K = \left((U_A \otimes U_B)\Pi^T\right)\left(\Pi(\Sigma_A \otimes \Sigma_B)\Pi^T\right)\left(\Pi(V_A \otimes V_B)^T\right),$$

*where $\Pi$ is a permutation matrix ($\Pi^T\Pi = I$) such that the singular values on the diagonal of $\Pi(\Sigma_A \otimes \Sigma_B)\Pi^T$ appear in non-increasing order.*

Proof. The validity is easily observed from the second property of Lemma 3.6 by inserting the SVDs of $A$ and $B$. □

Lemma 3.7 provides the basis for a formulation of filtered SVD solutions to $Ax = b^\delta$ where $A = A_2 \otimes A_1$. Let $A_1 = U_{A_1}\Sigma_{A_1}V_{A_1}^T$ and $A_2 = U_{A_2}\Sigma_{A_2}V_{A_2}^T$ be the SVDs of $A_1$ and $A_2$, then

$$x_\alpha = (V_{A_2} \otimes V_{A_1})\Pi^T\Phi_\alpha\Pi(\Sigma_{A_2}^\dagger \otimes \Sigma_{A_1}^\dagger)\Pi^T\Pi(U_{A_2} \otimes U_{A_1})^T b^\delta.$$

Note that we can use the first property from Lemma 3.6 to compute the solutions efficiently; e.g, $(U_{A_2} \otimes U_{A_1})^T b^\delta = U_{A_1}\mathrm{vec}^{-1}\left(b^\delta, m, n\right)U_{A_2}$ where $\mathrm{vec}^{-1}$ is defined in Definition 3.4. This approach can also be generalized to the case where the PSF matrix is approximated by a Kronecker product or possibly a sum of several Kronecker products, see, e.g., [69].

Without an SVD, we cannot construct TSVD solutions, but if the PSF matrix is still structured such that a diagonalization is possible, then we can directly compute

standard-form Tikhonov solutions. Assume, e.g., that $A \in \mathbb{R}^{m\times n}$ implements a nonseparable but invariant and symmetric PSF with reflexive boundary conditions. Table 3.1, shows that $A$ is then diagonizable by the 2D-DCT. Let $\Psi_1 \in \mathbb{R}^{m\times m}$ and $\Psi_2 \in \mathbb{R}^{n\times n}$ be one-dimensional DCT matrices as defined in Definition 3.1 and let $\Psi = \Psi_2 \otimes \Psi_1$ be a 2D-DCT matrix. Then $\Psi A \Psi^T = \Lambda$ is diagonal, and we can formulate the standard-form Tikhonov problem as

$$(\Lambda + \lambda^2 I)\Psi^T x_\lambda = \Lambda \Psi^T b^\delta,$$

and $x_\lambda$ can be computed using fast 2D-DCT transforms and diagonal matrix-vector products. If no diagonalization is possible, then we must consider iterative methods, which is the topic of several of the following chapters. Whether or not it is possible to directly formulate Tikhonov solutions to general-form problems depends on the structure of the regularization matrix $L$. These issues are studied in Chapter 5.

### 3.3.1 Numerical Example

We illustrate the impact of image regularization on a restoration problem. The problem is based on a structured PSF matrix such that Tikhonov solutions can be computed directly. Furthermore, an algorithm is constructed for computing the optimal Tikhonov parameter. The algorithm is schematically shown in Alg. 3.1.

**Algorithm 3.1**

***Location of optimal Tikhonov solution***
*A procedure for locating the optimal Tikhonov regularization parameter $\lambda_m$. The algorithm should be given a tolerance $\tau$ and initial $\lambda$ values. Moreover, $x_{\text{exact}}$ denotes the exact solution and $\#\mathcal{L}$ is the number of elements in $\mathcal{L}$.*

*Initialize list of $n = 3$; $\lambda$ values: $\mathcal{L} = \{\lambda_1, \lambda_2, \lambda_3\}$*
*Compute $x_{\lambda_i}$, $i = 1, 2, 3$*
*Set initial lambda error: $\Delta_\lambda = 1$*
*while* $\Delta_\lambda < \tau$
  *compute* $m = \text{argmin}_{i=1,\ldots,\#\mathcal{L}} \|x_{\text{exact}} - x_{\lambda_i}\|_2 / \|x_{\text{exact}}\|_2$
  *if* $m = 1$
    $\lambda_{\text{new}} = 10^{(\log_{10}(\lambda_1)+\log_{10}(\lambda_2))/2}$
    $\mathcal{L} = \{\lambda_1, \lambda_{\text{new}}, \lambda_2, \ldots, \lambda_n\}, \quad \Delta = |\lambda_2 - \lambda_1|$
  *else if* $m = n$
    $\lambda_{\text{new}} = 10^{(\log_{10}(\lambda_{n-1})+\log_{10}(\lambda_n))/2}$
    $\mathcal{L} = \{\lambda_1, \ldots, \lambda_{n-1}, \lambda_{\text{new}}, \lambda_n\}, \quad \Delta = |\lambda_n - \lambda_{n-1}|$
  *else*
    $\lambda_{\text{n1}} = 10^{(\log_{10}(\lambda_{m-1})+\log_{10}(\lambda_m))/2}$
    $\lambda_{\text{n2}} = 10^{(\log_{10}(\lambda_m)+\log_{10}(\lambda_{m+1}))/2}$
    $\mathcal{L} = \{\lambda_1, \ldots, \lambda_{m-1}, \lambda_{\text{n1}}, \lambda_m, \lambda_{\text{n2}}, \lambda_{m+1}, \ldots, \lambda_n\}$; $\Delta = |\lambda_{m+1} - \lambda_{m-1}|/2$

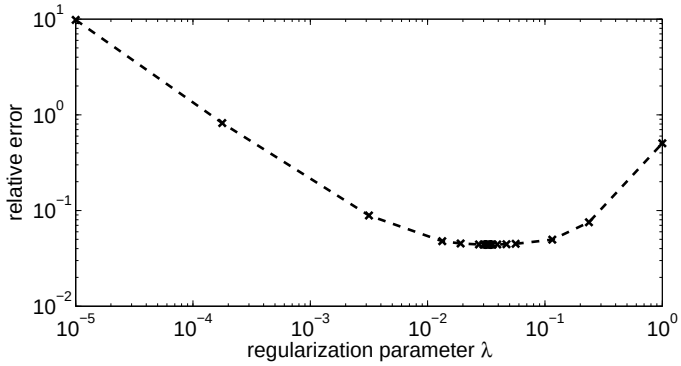



Figure 3.2: Relative errors of Tikhonov solutions $x_\lambda$ compared to the true solution $\mathrm{vec}\,(X_{\mathrm{exact}})$

**Example 3.2** *We consider a constructed image deblurring example such that the relative error $\varepsilon_2(x_\lambda)$ from* (2.17) *can be computed. The original image is $\bar{x}_{\mathrm{exact}} = \mathrm{vec}\,(\overline{X}_{\mathrm{exact}})$, and the blurred right-hand side is constructed as $\bar{b} = \overline{A}\bar{x}_{\mathrm{exact}}$ where $\overline{A}$ is a PSF matrix that implements atmospheric turbulence blur [38] as described in* (3.9) *with $\sigma_s = 4$ and $\sigma_t = 6$. To avoid boundary effects in the true blurred image, then the outermost 50 pixels are cut off, resulting in the blurred image $B \in \mathbb{R}^{597\times 593}$. Noise is added such that $b^\delta = \mathrm{vec}\,(B) + e$ where $e \in \mathbb{R}^{354021}$ is white Gaussian noise, scaled such that $\|e\|_2/\|\mathrm{vec}\,(B)\,\|_2 = 10^{-6}$.*

*We want to reconstruct the solution $X_{\mathrm{exact}}$, i.e., $\overline{X}_{\mathrm{exact}}$ where again the outermost 50 pixels have been removed. The PSF matrix $A \in \mathbb{R}^{354021\times 354021}$ describes atmospheric turbulence blur, and reflexive boundary conditions are chosen such that Tikhonov solutions can be computed directly using 2D-DCTs. The top row of Fig. 3.3 shows the true solution $X_{\mathrm{exact}}$ and an image of the PSF.*

*We use Alg. 3.1 to find the optimal Tikhonov regularization parameter and the optimal Tikhonov solution. Figure 3.2 shows the the relative errors as a function of $\lambda$, and the optimal $\lambda$ is $3.2859 \cdot 10^{-2}$.*

*The bottom row of Fig. 3.3 shows the blurred and noisy image $B^\delta$ as well as the optimal Tikhonov solution. We note that the reconstructed image is deblurred considerably compared to $B^\delta$. For more on the quality of deblurred images, see e.g., [52].*

## 3.4 Summary

In this chapter, we studied the matrix structures that arise naturally in the formulation of image deblurring problems. These structures should be exploited in the efficient implementation of e.g., matrix-vector products. Furthermore, Table 3.1

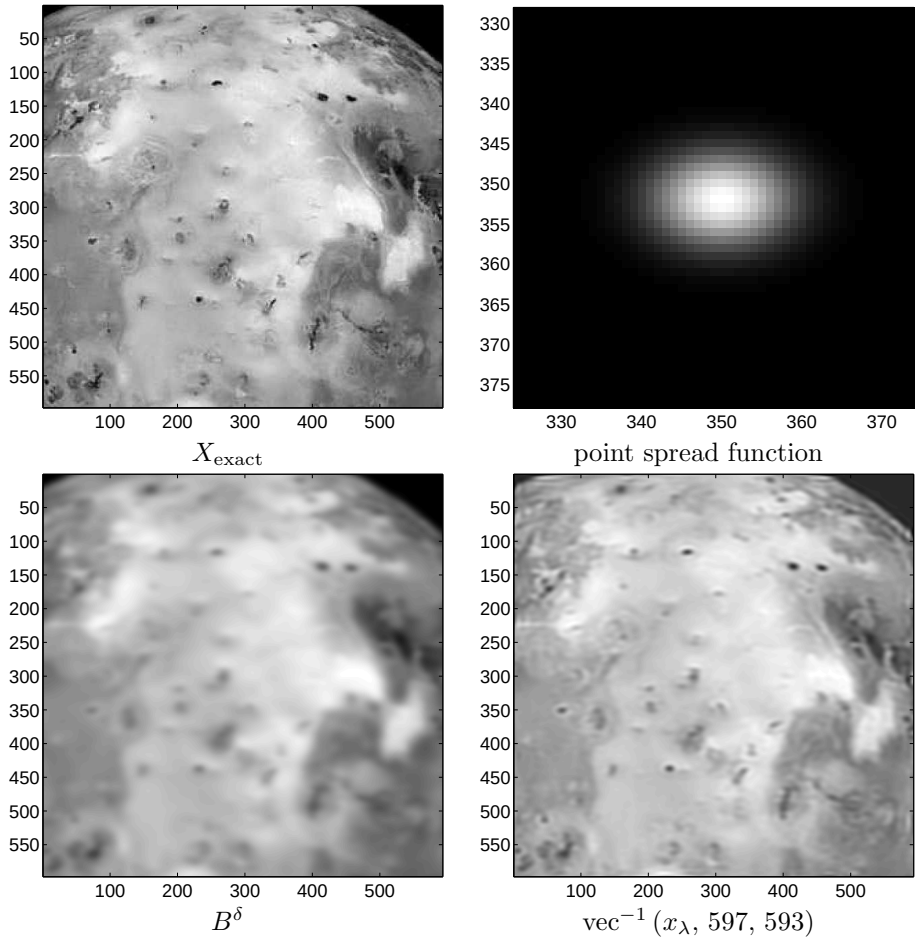



Figure 3.3: True image, PSF image, blurred image, and optimal Tikhonov solution to the image deblurring example.

was generated to provide an overview over corresponding properties of the PSF and the PSF matrices. Finally, we showed an example of an image deblurring problem where the boundary conditions were chosen such that Tikhonov solutions could be computed directly.

Chapter 4

# Iterative Regularization Methods

In this chapter, we will look deeper into some Krylov subspace methods and especially study the regularization properties when applying these methods directly to the problems $Ax = b$ or $\min_x \|b - Ax\|_2$. The Krylov subspace methods are projection methods and can be described as special cases of the more general subspace correction methods. A general projection method seeks a solution $x_k \in \mathcal{W}_k + x_0$ in the (possibly affine) subspace $\mathcal{W}_k + x_0$ for which the residual $r_k = b - A\,x_k$ is orthogonal to another suitable $k$-dimensional subspace $\mathcal{V}_k$, i.e., $r_k \perp \mathcal{V}_k$. These orthogonality constraints are called the Petrov-Galerkin conditions. The resulting methods are Krylov methods if the subspaces $\mathcal{V}_k$ and $\mathcal{W}_k$ are generated as Krylov sequences. For more on Krylov methods in general, see e.g., [20, 28, 48, 86].

## 4.1 General Krylov Subspace Methods

The Krylov subspaces provide the basis for the iterative methods studied here, and we therefore start with the definition of a Krylov subspace.

**Definition 4.1** (Krylov Subspace) Given a matrix $A \in \mathbb{R}^{n\times n}$, a vector $r \in \mathbb{R}^n$, and an integer $k = 1, \ldots, n$, the Krylov subspace $\mathcal{K}_k(A, r)$ is defined by

$$\mathcal{K}_k\,(A, r) = \operatorname{span}\left\{r, Ar, \ldots, A^{(k-1)}r\right\}.$$

Along with Definition 4.1, Krylov subspaces have a number of important properties. Some of these are stated in the following Lemma.

**Lemma 4.2** *Let the Krylov subspace $\mathcal{K}_k(A, r)$ be defined as in Definition 4.1, and let* $\dim(\cdot)$ *denote the dimension of a subspace. Then the following statements hold*

1. $\dim(\mathcal{K}_k(A, r)) \leq k$
2. $\dim(A\mathcal{K}_k(A, r)) \leq \dim(\mathcal{K}_k(A, r))$
3. $\dim(\mathcal{K}_k(A, r)) \leq \dim(\mathcal{K}_{k+1}(A, r))$

The statements in Lemma 4.19 are important for identifying breakdown and stagnation of the Krylov subspace methods.

### 4.1.1 OR and MR Approaches

We discuss the Orthogonal Residual (OR) and Minimum Residual (MR) approaches in connection with Krylov subspaces for solving linear systems of equations. First, let $S \in \mathbb{R}^{n\times n}$ be an invertible matrix, let $y$, $z \in \mathbb{R}^n$ be two vectors, and define the linear system

$$Sy = z. \tag{4.1}$$

We start an iterative solution process by choosing an initial solution guess $y_0$, which defines the initial residual vector $r_0 = z - S\,y_0$. Then we approximate the solution to (4.1) by iteratively finding corrections $c^{(k)}$ to $y_0$ in Krylov subspaces $\mathcal{K}_k(S, r_0)$ of increasing dimension. The solutions therefore lie in the affine subspace $\mathcal{K}_k(S, r_0)+y_0$, i.e, $y^{(k)} = y_0 + c^{(k)} \in \mathcal{K}_k(S, r_0) + y_0$, and the residual is $r^{(k)} = z - Sy^{(k)} = r_0 - Sc^{(k)}$ where $c^{(k)} \in \mathcal{K}_k(S, r_0)$.

We need a way to determine the corrections $c^{(k)}$ from the Krylov subspaces, and normally two different approaches are considered. The OR approach determines $c^{(k)}_{\mathrm{OR}}$ as follows

$$\text{find} \quad c^{(k)}_{\mathrm{OR}} \in \mathcal{K}_k(S, r_0) \quad \text{such that} \quad r_0 - Sc^{(k)}_{\mathrm{OR}} \perp \mathcal{K}_k(S, r_0) \tag{4.2}$$

i.e., it seeks the correction $c^{(k)}_{\mathrm{OR}}$ in the subspace $\mathcal{K}_k(S, r_0)$ such that the corresponding residual is orthogonal to the same subspace. The Petrov-Galerkin conditions are in this case generally called just the Galerkin conditions.

The MR approach, on the other hand, determines $c^{(k)}_{\mathrm{MR}}$ by minimizing the 2-norm of the residual, i.e.,

$$\text{find} \quad c^{(k)}_{\mathrm{MR}} \in \mathcal{K}_k(S, r_0) \quad \text{such that} \quad \|r_0 - Sc^{(k)}_{\mathrm{MR}}\|_2 \quad \text{is minimized.} \tag{4.3}$$

Alternatively, the MR approach can also be described using the Petrov-Galerkin conditions by

$$\text{find} \quad c^{(k)}_{\mathrm{MR}} \in \mathcal{K}_k(S, r_0) \quad \text{such that} \quad r_0 - Sc^{(k)}_{\mathrm{MR}} \perp S\mathcal{K}_k(S, r_0). \tag{4.4}$$

The equivalence of the two formulations (4.3)–(4.4) is shown in the following section.

For both the OR approach and the MR approach, we choose the corrections $c_{\mathrm{OR}}^{(k)}$ and $c_{\mathrm{MR}}^{(k)}$ from the same subspace $\mathcal{K}_k(S, r_0)$, and looking at the residuals, we see that they also belong to identical Krylov subspaces, namely $r^{(k)} \in S\mathcal{K}_k(S, r_0) + r_0 = \mathcal{K}_{k+1}(S, r_0)$. Therefore, both approaches define two connected Krylov subspaces – the *correction subspace* $\mathcal{K}_k(S, r_0)$, and the *residual subspace* $\mathcal{K}_{k+1}(S, r_0)$.

Often, the initial solution iterate is set to zero, $y_0 = 0$, in which case the initial residual becomes $r_0 = z - Sy_0 = z$, and $y^{(k)} \in \mathcal{K}_k(S, r_0) = \mathcal{K}_k(S, z)$. In this case the correction subspace is also called the *solution subspace.* In the rest of this thesis we shall assume that $y_0 = 0$.

Krylov subspaces are generated by successive multiplications with the coefficient matrix, and can therefore be conveniently expressed by matrix polynomials. Any solution iterate $y^{(k)} \in \mathcal{K}_k(S, z)$ can therefore be given as

$$y^{(k)} = \sum_{i=0}^{k-1} \gamma_i S^i z = \mathcal{P}_k(S)z, \tag{4.5}$$

where $\mathcal{P}_k$ is a polynomial not exceeding degree $k-1$, and $\gamma_i$ is the $i$th polynomial coefficient. We know from above that the residual also lies in a Krylov subspace, and we see that the residual must satisfy

$$r^{(k)} = z - Sy^{(k)} = (I - S\mathcal{P}_k(S))\, z = \mathcal{Q}_k(S)z, \tag{4.6}$$

where $\mathcal{Q}_k$ is called the residual polynomial. It is a polynomial not exceeding degree $k$ and as seen with the special property that $\mathcal{Q}_k(0) = 1$. Any Krylov method therefore defines a solution polynomial $\mathcal{P}_k$ and a corresponding residual polynomial $\mathcal{Q}_k$. The residual polynomial also defines the error $y^* - y^{(k)}$ where $y^*$ is the exact solution to the system (4.1):

$$y^* - y^{(k)} = y^* - \mathcal{P}_k(S)z = y^* - \mathcal{P}_k(S)Sy^* = \mathcal{Q}_k(S)y^*. \tag{4.7}$$

The coefficients of the polynomials are fixed by the Galerkin or Petrov-Galerkin conditions (4.2)–(4.4) for OR and MR approaches, respectively.

A natural way to analyze the Krylov methods is by means of the eigenvalue decomposition, if this exists. Let $S = W\Lambda W^{-1}$ be an eigenvalue decomposition. Then the solution iterates and the residuals can be written as

$$y^{(y)} = W\mathcal{P}_k(\Lambda)W^{-1}z \tag{4.8}$$

$$r^{(k)} = W\mathcal{Q}_k(\lambda)W^{-1}z, \tag{4.9}$$

i.e., the polynomials can be described to act as filters in the eigenvalues. This approach is often used as a means for bounding the relative residual. A worst-case bound on the relative residual is e.g., given by [86, Proposition 6.32]

$$\frac{\|r^{(k)}\|_2}{\|r_0\|_2} = \frac{\|W\mathcal{Q}_k(\lambda)W^{-1}z\|_2}{\|z\|_2} \leq \min_{p\in\Pi_k} \max_i \kappa(W)|p(\lambda_i)|,$$

where $\Pi_k$ denotes the set of all polynomials $\pi_k$ not exceeding degree $k$ for which $\pi_k(0) = 1$, and $\kappa(W)$ is the condition number of the eigenvector matrix. Obviously, if

$\kappa(W)$ is large, the bound might indeed be bad, and for more general nondiagonizable matrices the analysis of the Krylov methods is less well understood. Note that the expression simplifies somewhat if $S$ is normal, because in this case $W$ is unitary and therefore $\kappa(W) = 1$. We refer to e.g., [23, 47, 57, 64, 63, 87, 90, 94] and references therein for in-depth analyses of several different Krylov subspace methods.

There exists a vast quantity of algorithms and implementations that are based on the two general Krylov schemes above. Among the OR methods are the famous CG (conjugate gradient), the FOM (full orthogonalization method), CGLS (CG applied implicitly to the normal equations), LSQR (equivalent to CGLS, but implemented using Lanczos bidiagonalization), etc. To the class of MR methods, the most general method is GMRES (general minimum residual method) that solves problems with general, possibly nonsymmetric, coefficient matrices. For symmetric, but possibly indefinite coefficient matrices, the mathematically equivalent method MINRES is a clever implementation based on the Lanczos tridiagonalization algorithm. The list continues and can be extended with more exotic variants and approximations such as QRM, BiCG, CGS, GMRES($m$), etc. All these algorithms try either to improve the numerical stability or decrease the computational work or needed memory storage. For example, GMRES needs to carry along an explicit basis for the generated Krylov subspaces of increasing size. QMR and GMRES($m$) try to deal with this issue by either constructing a non-optimal basis and minimizing a quasi-residual, or by restarting GMRES for each $m$ iterations, using a new clever starting guess. For more complete surveys, see e.g., [23, 28, 86].

In the following, we will study the minimum-residual methods GMRES and MINRES, as well as the variants RRGMRES [7] and MR-II [31] developed for solving inconsistent systems. We will also compare with the more widely used CGLS, conjugate gradients for the least squares problem.

### 4.1.2 Basic OR and MR Algorithms

Let us first describe basic OR and MR algorithms using a common framework. In both cases, we must construct the canonical Krylov bases that make up the solution and residual subspaces. First, we define the linear system

$$Ax = b, \quad A \in \mathbb{R}^{n\times n}, \quad x,\, b \in \mathbb{R}^n, \tag{4.10}$$

where $A$ is now a general square matrix, and we assume that the initial solution iterate is $x^{(0)} = 0$. Therefore $r^{(0)} = b - Ax^{(0)} = b$, and the solution subspace is chosen to be the standard Krylov subspace

$$\mathcal{K}_k(A, b) = \left\{b,\, Ab,\, A^2b,\, \ldots,\, A^{k-1}b\right\}. \tag{4.11}$$

We apply Arnoldi's method to construct orthogonal bases for both the solution subspace $\mathcal{K}_k(A, b) = \operatorname{span}\{w_1, w_2, \ldots, w_k\}$ and the residual subspace $\mathcal{K}_{k+1}(A, b) = \operatorname{span}\{w_1, w_2, \ldots, w_{k+1}\}$. In Alg. 4.1, Arnoldi's method is described such that orthogonalization of the Krylov bases are achieved by modified Gram-Schmidt, i.e., the columns of the matrices $W_k \in \mathbb{R}^{n\times k}$ and $W_{k+1} \in \mathbb{R}^{n\times(k+1)}$ span the solution and

residual subspaces. Alternatively, Arnoldi's method can also be implemented using, e.g., Householder orthogonalization. Arnoldi's method gives rise to two partial decompositions

$$W_k^T A W_k = H_k \tag{4.12}$$

$$A W_k = W_{k+1} \overline{H}_k, \tag{4.13}$$

where $H_k \in \mathbb{R}^{k \times k}$ is a square upper Hessenberg matrix, and $\overline{H}_k \in \mathbb{R}^{(k+1)\times k}$ is a rectangular upper Hessenberg. Furthermore, $W_{k+1}\overline{H}_k = W_k H_k + w^{(k)} e_k^T$, i.e., $W_{k+1}\overline{H}_k$ is a rank-one update of $W_k H_k$, and the upper square part of $\overline{H}_k$ is $H_k$.

**Algorithm 4.1**

***Arnoldi's Method***
*Schematic illustration of Arnoldi's method with orthogonalization of the Krylov vectors implemented using modified Gram-Schmidt.*

1. $\rho = \|r^{(0)}\|_2$
2. $w_1 = b/\rho$
3. ***for*** $j = 1$ *to* $k$
4. $\quad d^{(j)} = A w_j$
5. $\quad$ ***for*** $i = 1$ *to* $j$
6. $\quad\quad H_{i,j} = d^{(j)T} w_i$
7. $\quad\quad d^{(j)} = d^{(j)} - H_{i,j} w_i$
8. $\quad H_{j+1,j} = \|d^{(j)}\|_2$
9. $\quad w_{j+1} = d^{(j)} / H_{j+1,i}$

To define the OR solution $\bar{x}^{(k)}$, we apply the Galerkin conditions (4.2) using the matrix $W_k$ as the basis for the solution subspace, and we get

$$b - A\bar{x}^{(k)} \perp \mathcal{K}_k(A, b) \quad \Leftrightarrow \quad b - A W_k \bar{\xi}_k \perp \mathcal{R}(W_k),$$

where $\bar{x}^{(k)} = W_k \bar{\xi}_k$. This is equivalent to

$$W_k^T (b - A W_k \bar{\xi}_k) = 0 \quad \Leftrightarrow \quad W_k^T A W_k \bar{\xi}_k = W_k^T b \quad \Leftrightarrow \quad H_k \bar{\xi}_k = \rho e_1,$$

where $e_1 = (1, 0, \ldots, 0)^T$ denotes the first canonical unit vector, and $\rho = \|b\|_2$. The last identity follows from (4.12), and the fact that the first column of $W_k$ in Alg. 4.1 is the normed right-hand side $w_1 = b/\|b\|_2$, and that all subsequent columns by construction are orthogonal to this.

The projected system is now of dimensions $k \times k$ and the OR solution is found easily by

$$\bar{x}^{(k)} = W_k \bar{\xi}_k, \quad \text{where} \quad \bar{\xi}_k = H_k^{-1} \rho e_1 \tag{4.14}$$

provided that the Hessenberg matrix $H_k$ is nonsingular.

The MR solution can be defined in a similar way by imposing the Petrov-Galerkin conditions (4.4)

$$b - Ax^{(k)} \perp A\mathcal{K}_k(A, b) \quad \Leftrightarrow \quad b - AW_k\xi_k \perp \mathcal{R}(AW_k).$$

Using the relation (4.13) the orthogonality conditions can be restated as

$$W_k^T A^T (b - AW_k\xi_k) = 0 \Leftrightarrow \overline{H}_k^T \overline{H}_k \xi_k = \overline{H}_k^T \rho e_1,$$

and the solution to the projected system is again easily found by

$$x^{(k)} = W_k\xi_k, \quad \text{where} \quad \xi_k = (\overline{H}_k^T \overline{H}_k)^{-1} \overline{H}_k^T \rho e_1, \tag{4.15}$$

given that $\overline{H}_k^T \overline{H}_k$ is nonsingular. We observe that $\overline{H}_k^\dagger = (\overline{H}_k^T \overline{H}_k)^{-1} \overline{H}_k^T$ is a Moore-Penrose pseudoinverse and that $\xi_k$ is the least squares solution of minimum norm, i.e., the unique minimizer of $\|\rho e_1 - \overline{H}_k \xi\|_2 = \|b - AW_k\xi\|_2$. This shows the equivalence of the two formulations of the MR approach in (4.3)–(4.4).

The conditions for $H_k$ and $\overline{H}_k^T \overline{H}_k$ to be nonsingular are e.g., described in [86, Proposition 5.1]. To guarantee the existence of the OR iterates, $A$ must be positive definite, and to guarantee the existence of the MR iterates, $A$ must be nonsingular. This shows that the OR approach is only applicable to positive definite problems, whereas the MR iterates are well-defined also for indefinite problems.

General representatives of algorithms implementing the OR approach and the MR approach, respectively, are the *full orthogonalization method* (FOM) and the *generalized minimum residual* (GMRES).

### 4.1.3 Symmetric Systems

Now assume that $A \in \mathbb{R}^{n\times n}$ is symmetric. By applying Arnoldi's method, we still get the partial decompositions (4.12) and (4.13), but it is easily observed that (4.12) implies that $H_k$ in this case is also symmetric. By construction $H_k$ is still of Hessenberg form, which results in $H_k$ being symmetric and tridiagonal. This simpler structure can be exploited in several ways. For instance, the partial decompositions can be performed by the simpler Lanczos method which is the basis for the MR algorithm MINRES [78]. Furthermore, the tridiagonal structure leads to simple three-term recurrence relations such that the MINRES solution iterates can be updated without storing the entire partial decomposition. Also one of the best known iterative methods for solving large sparse symmetric and positive definite systems of equations, the *conjugate gradient* (CG) algorithm, produces OR iterates via a clever update structure. This algorithm is equivalent to using a Cholesky factorization of the tridiagonal matrix arising from Arnoldi's method or the Lanczos process [79], and produces solutions for which the $A$-norm of the error $x^* - x$ is minimized:

$$x_{\mathrm{CG}}^{(k)} = \mathrm{argmin}_x \|x^* - x\|_A, \quad x \in \mathcal{K}_k(A, b), \tag{4.16}$$

where $x^*$ is the exact solution of (4.10). The $A$-norm is here well-defined because CG is only applicable to systems where $A$ is symmetric and positive definite.

### 4.1.4 The Normal Equations

If $A \in \mathbb{R}^{m\times n}$ is rectangular then none of the above approaches apply because the construction of the Krylov subspaces fail. Furthermore, even if $m = n$ and $A$ is indefinite, the OR approach is not guaranteed to be well-defined, and indeed if $A$ is nonsymmetric, no short recurrence relations apply for constructing the Krylov subspaces. In any of these cases, it might be useful to look at the corresponding normal equations

$$A^T Ax = A^T b \qquad \text{(overdetermined systems) or} \tag{4.17}$$

$$AA^T y = b, \quad x = A^T y \quad \text{(underdetermined systems),} \tag{4.18}$$

where the matrices $A^T A$ and $AA^T$ are symmetric and positive (semi-)definite. We primarily look at systems described by (4.17). First, the Krylov subspace is in this case well-defined and given by

$$\mathcal{K}_k\left(A^T A, A^T b\right) = \operatorname{span}\left\{A^T b, (A^T A)A^T b, \ldots, (A^T A)^{k-1} A^T b\right\}, \tag{4.19}$$

and by applying Arnoldi's method or an equivalent symmetric Lanczos method, we get again two partial decompositions

$$\overline{W}_k^T A^T A \overline{W}_k = T_k \tag{4.20}$$

$$A^T A \overline{W}_k = \overline{W}_{k+1} \overline{T}_k, \tag{4.21}$$

where $\overline{W}_k = (\,\bar{w}_1, \bar{w}_2, \ldots, \bar{w}_k\,)$ such that $\mathcal{K}_k\left(A^T A, A^T b\right) = \operatorname{span}\{\bar{w}_1, \bar{w}_2, \ldots, \bar{w}_k\}$, and $T_k \in \mathbb{R}^{k\times k}$ is a tridiagonal matrix equivalent to the Hessenberg matrix in (4.12). Similarly, $\overline{T}_k \in \mathbb{R}^{(k+1)\times k}$ is also tridiagonal with an additional element in the lower right corner. Again, we can fix the OR solution by applying the Galerkin conditions. We get

$$r^{(k)} = A^T b - A^T A x^{(k)} \perp \mathcal{K}_k(A^T A, A^T b) \Leftrightarrow$$

$$\overline{W}_k^T (A^T b - A^T A \overline{W}_k \xi_k) = 0 \quad \Leftrightarrow \quad \min \|b - A\overline{W}_k \xi_k\|_2, \quad \xi_k \in \mathbb{R}^k, \tag{4.22}$$

and we observe that imposing the Galerkin conditions here implies that the 2-norm of the residual $b - Ax$ is minimized for $x \in \mathcal{K}_k(A^T A, A^T b)$. Thus the Galerkin conditions in combination with the normal equations from (4.17) yields a special minimum-residual method for the original problem. The only difference is that the solution lies in a different subspace.

We can of course also apply the Petrov-Galerkin conditions for solving the normal equations and arrive at a method that minimizes the residual of the normal equations. I.e., we get the solution iterates that satisfy

$$\min \|A^T b - A^T A x^{(k)}\|_2, \quad x \in \mathcal{K}_k(A^T A, A^T b).$$

Using the other set of normal equations (4.18) does not in general result in an optimality criterion like the minimized residual for the formulation (4.17). For consistent problems where $b \in \mathcal{R}(A)$ then the error is minimized over the Krylov subspace, i.e., $\min \|x^\dagger - x_k\|_2$ where $x^\dagger$ is the least squares solution to the system. A method implementing this approach is known as Craig's method, see e.g., Hanke [31] and references therein.

In the following, we will look at algorithms used for computing regularized solutions to rank-deficient and discrete ill-posed problems, and later on the regularizing properties of these algorithms.

## 4.2 Methods for Inverse Problems

We now want to compute regularized solutions by applying Krylov methods directly to $Ax = b$ or $\min_x \|b - Ax\|_2$. In doing so, it is of great importance that the algorithms provide suitable solution subspaces, and not only a fast decreasing residual norm. We saw in Chapter 2 that to define a regularization method theoretically, we need both a regularized operator and a suitable regularization parameter. Using Krylov methods the hope is that the solution polynomials act as regularized operators, and that the number of iterations $k$ act as a discrete regularization parameter.

In Section 4.1.1 we mentioned briefly that convergence analyses and bounds on the residual are often formulated by means of the eigenvalues. Now we are not particularly interested in the overall reduction of the residual, but far more interested in how the iterates of a Krylov method applied to a rank-deficient or discrete ill-conditioned problem approximates the wanted underlying solution. As we saw in Chapter 2, the two well-known regularization methods TSVD and Tikhonov regularization are based on computing regularized approximations to the least squares solutions of minimum norm. Furthermore, they are both defined by filters in the SVD domain. For a general Krylov method to provide regularized solutions comparable to the TSVD and Tikhonov solutions, there must be some correspondence between a possible eigenbasis and the SVD basis.

### 4.2.1 Using the Normal Equations

As mentioned earlier, it is a natural idea to form the normal equations (4.17) or (4.18) if the system to solve is either over- or underdetermined or indefinite. And because the coefficient matrices $A^TA$ and $AA^T$ are symmetric and positive (semi-)definite, it is just as natural to apply CG to these systems, which is the classical approach for regularizing iterations.

We apply CG only to the system (4.17) to obtain a minimization of the residual as described in Section 4.1.4. Unfortunately, several names are used to describe this algorithm. In general, when applying the CG method to the two normal equations systems (4.17) or (4.18), the resulting algorithms are often called CGNR and CGNE, see e.g., Saad [86]. The $N$ after the CG indicates that we apply CG to the normal

equations, and the following $R$ or $E$ indicate that we implicitly minimize the 2-norm of the residual or the 2-norm of the error for the normal equations. See e.g., Gutknecht [30, Theorems 2.9.1 and 2.9.2] for the Galerkin approximations obtained for the two normal equations approaches. Other authors, e.g., Engl et al. [22], use CGNE to indicate both cases (4.17) and (4.18). Furthermore, Hanke [31] introduces the name CGME for CG applied directly to (4.18).

Here we use the name CGLS – CG for least squares – to describe the algorithm that applies CG to (4.17). Moreover, a different implementation that is mathematically equivalent to CGLS is the LSQR algorithm due to Paige and Saunders [79]. This algorithm is also used in the following as it better allows for an analysis of the properties of the methods because it explicitly constructs a basis for the involved Krylov subspaces. It is based on the Lanczos bidiagonalization scheme bidiag1 [25], see Alg. 4.2.

**Algorithm 4.2**

***Lanczos Bidiagonalization***
*Schematic illustration of the Lanczos bidiagonalization method.*

1. $r_0 = b$
2. $\beta_1 = \|r_0\|_2$
3. $\bar{u}_1 = r_0/\rho$
4. $\bar{w}_0 = 0$
5. ***for*** $j = 1$ *to* $k$
6. $\quad r_j = A^T \bar{u}_j - \beta_j \bar{w}_{j-1}$
7. $\quad \alpha_j = \|r_j\|_2$
8. $\quad \bar{w}_j = r_j/\alpha_j$
9. $\quad q_j = A\bar{w}_j - \alpha_j \bar{u}_j$
10. $\quad \beta_{j+1} = \|q_j\|_2$
11. $\quad \bar{u}_{j+1} = q_j/\beta_{j+1}$

Both CGLS and LSQR work *implicitly* with the normal equations without ever forming the matrix $A^T A$ which is the basis for the used Krylov subspace (4.19). Application of the Lanczos bidiagonalization algorithm from Alg. 4.2 gives the partial decomposition

$$A\overline{W}_k = \overline{U}_{k+1} B_k, \tag{4.23}$$

where $\overline{W}_k \in \mathbb{R}^{n\times k}$ has orthonormal columns that span the Krylov subspace (4.19). The matrix $\overline{U}_{k+1} \in \mathbb{R}^{n\times(k+1)}$ also has orthonormal columns, and its first column is

$u_0 = b/\|b\|_2$. The matrix $B_k \in \mathbb{R}^{(k+1)\times k}$ is a lower bidiagonal matrix of the form

$$B_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix},$$

where the $\alpha$s and $\beta$s are from Alg. 4.2. Indeed, we see that this is similar to applying Arnoldi's method to the coefficient matrix $A^TA$, because by transposing (4.23), we get $\overline{W}_k^T A^T = B_k^T \overline{U}_{k+1}^T$ such that

$$\overline{W}_k^T A^T A \overline{W}_k = B_k^T \overline{U}_{k+1}^T \overline{U}_{k+1} B_k = B_k^T B_k,$$

is similar to (4.12) and $B_k^T B_k$ is tridiagonal.

Using the Galerkin conditions (4.22) as well as (4.23), it follows that

$$\bar{x}^{(k)} = \overline{W}_k \xi_k, \qquad \bar{\xi}_k = \operatorname{argmin}_{\bar{\xi}\in\mathbb{R}^k} \|B_k \bar{\xi} - \rho e_1\|_2, \tag{4.24}$$

where $e_1$ is the first canonical unit vector in $\mathbb{R}^{k+1}$ and $\rho = \|b\|_2$. This algorithm can be implemented using short recurrences and thus one can avoid storing the partial decomposition (4.23).

We study the Krylov subspaces that make up the basis for the solutions. In this case, $A^TA$ is diagonizable, and the eigenvalues are the squared singular values. Therefore, we get the following

**Lemma 4.3** *Using Krylov subspace methods based on the Krylov subspace* (4.19) *generates solution iterates that can be described by a spectral filtering in the SVD sense. Let the SVD of the matrix $A$ be given as $A = U\Sigma V^T$. Solution components $V^T x^{(k)}$ computed from the Krylov subspace* (4.19) *are then given by*

$$V^T \bar{x}^{(k)} \in \operatorname{span}\left\{\Sigma U^T b, \Sigma^3 U^T b, \ldots, \Sigma^{2k-1} U^T b\right\} = \mathcal{K}_k\left(\Sigma^2, \Sigma U^T b\right), \tag{4.25}$$

*and the solutions are then easily expressed in terms of the SVD of $A$ as*

$$\bar{x}^{(k)} = V\,\overline{\Phi}_k\,\Sigma^\dagger U^T b, \qquad \overline{\Phi}_k = \mathcal{P}_k(\Sigma^2)\Sigma^2,$$

*where $\overline{\Phi}_k$ is a diagonal matrix and $\overline{\mathcal{P}}_k$ is a solution polynomial not exceeding degree $k-1$.*

Proof. First we note that $(A^TA)^k = V\Sigma^{2k}V^T$ and $(A^TA)^kA^T = V\Sigma^{2k+1}U^T$. Using these relations, Eq. (4.25) follows directly from expressing the Krylov subspace (4.19) in terms of the SVD of $A$ as

$$\mathcal{K}_k\left(A^TA, A^Tb\right) = \operatorname{span}\left\{V\Sigma U^T b, V\Sigma^3 U^T b, \ldots, V\Sigma^{2k-1}U^T b\right\}.$$

Because of the diagonal form of $\overline{\Phi}_k = \overline{\mathcal{P}}_k(\Sigma^2)\Sigma^2$, the solution can be described by filter factors in the SVD domain (2.13). □

Lemma 4.3 shows that Krylov methods based on the normal equations will generate solutions that can be described as filtered SVD solutions, similar to the TSVD and Tikhonov solutions.

### 4.2.2 Avoiding the Transposed Matrix

Now imagine that the transposed operator $A^T$ is not available. Suppose the operator $A$ is given only as a black-box function that, when given a vector $b$, returns the matrix-vector product $Ab$. In this case, it is not possible to form the normal equations. Therefore, we look at methods based directly on Arnoldi's method for generating a basis for the Krylov subspace $\mathcal{K}_k(A, b)$. These approaches are naturally limited to square matrices $A \in \mathbb{R}^{n\times n}$. Comparing to the case using the normal equations, this has the potential advantage that only one matrix-vector multiplication is needed in each iteration, whereas the normal equations case needs a multiplication with both $A$ and $A^T$.

If the system is positive definite then we can apply a Galerkin method, and if the system is also symmetric, we can apply standard CG. But in general, the system might very well be indefinite, and furthermore, as we will see later, we want to consider methods that minimize the residual. An obvious approach is to apply an MR algorithm to the original problem $Ax = b$.

**Algorithm 4.3**

***GMRES***
*Schematic illustration of the GMRES (general minimum-residual) algorithm.*

1. $d^{(0)} = Ax^{(0)} - b$
2. $\rho = \|d^{(0)}\|_2$
3. ***for*** $k = 1$ *to* $m$
4. $\quad w_k = d^{(k-1)}/H_{k+1,k}$
5. $\quad d^{(k)} = Aw_k$
6. $\quad$ ***for*** $i = 1$ *to* $k$
7. $\quad\quad H_{i,k} = d^{(k)T} w_i$
8. $\quad\quad d^{(k)} = d^{(k)} - H_{i,k} w_i$
9. $\quad H_{k+1,k} = \|d^{(k)}\|_2$
10. $\quad \xi_k = \operatorname{argmin}_y \|H\xi_k - \rho I\|_2$
11. $\quad x^{(k)} = x_0 + W_k \xi_k$

The method of choice for general nonsymmetric matrices is the GMRES method that constructs the partial decomposition (4.13) and follows the general MR approach (4.15). This procedure is schematically shown in Alg. 4.3. An actual implementation

is often based on a QR factorization of the generated Hessenberg matrix $\overline{H}_k$ that can be updated for each iteration. In this way the solution and the residual can also be updated for each iteration. If $A$ is symmetric then MINRES is a more elegant implementation, exploiting the fact that the Hessenberg matrix $\overline{H}_k$ reduces to tridiagonal form. In this case the solution can be updated without explicitly storing the partial decomposition (4.13). In the general case, no such short recurrences exist, and GMRES needs to carry along all the constructed Krylov vectors.

Variants of GMRES and MINRES exist that use $Ab$ as the starting vector for the Krylov subspace instead of $b$ as in the standard Krylov subspace (4.11). This gives the Krylov subspace

$$\mathcal{K}_k\left(A, Ab\right) = \text{span}\left\{Ab, A^2b, A^3b, \ldots, A^kb\right\}. \tag{4.26}$$

These methods are constructed to compute solutions to inconsistent systems. If $b \notin \mathcal{R}(A)$ then the standard Krylov subspace will not be a subspace in $\mathcal{R}(A)$ whereas indeed $\mathcal{K}_k\left(A, Ab\right) \in \mathcal{R}(A)$. In the symmetric case an efficient short recurrence implementation of a method that minimize the residual with respect to the subspace (4.26) is the MR-II algorithm [24, 31, 34]. In the non-symmetric case the algorithm RRGMRES exists [7, 8]. The partial decomposition when using MR-II or RRGMRES is denoted

$$A\,\widehat{W}_k = \widehat{W}_{k+1}\widehat{H}_k, \qquad \widehat{W}_{k+1} = (\,\widehat{W}_k\,,\,\hat{w}_{k+1}\,), \tag{4.27}$$

where $\widehat{W}_k \in \mathbb{R}^{n\times k}$ provides a basis for (4.26). Note that one extra matrix-vector multiplication is needed to obtain a Krylov subspace of the same dimension as standard GMRES/MINRES.

Let us look at these Krylov subspaces in the SVD basis. First define

$$A = U\Sigma V^T \quad , \quad C = V^TU \quad , \quad \beta = U^Tb, \tag{4.28}$$

i.e., the SVD of $A$, the matrix $C$ that expresses the left singular vectors in the right singular basis and the vector $\beta$ that expresses the right-hand side in the left singular basis. Now look at the Krylov subspaces.

**Lemma 4.4** *Krylov subspace methods based on the Krylov subspaces* (4.11) *or* (4.26) *applied to nonsymmetric matrices are in general* not *spectral filtering methods in the SVD sense. Using* (4.28)*, we describe the solution components* $V^Tx^{(k)}$ *and* $V^T\hat{x}^{(k)}$ *from the Krylov subspaces* (4.11) *and* (4.26) *as*

$$V^Tx^{(k)} \in \mathcal{K}_k\left(C\Sigma, C\beta\right) \quad , \quad V^T\hat{x}^{(k)} \in \mathcal{K}_k\left(C\Sigma, C\Sigma C\beta\right), \tag{4.29}$$

*and the solutions can then be expressed by*

$$x^{(k)} = V\,\Phi_k\,\Sigma^\dagger\beta, \qquad \Phi_k = \mathcal{P}_k(C\,\Sigma)C\,\Sigma, \tag{4.30}$$

$$\hat{x}^{(k)} = V\,\widehat{\Phi}_k\,\Sigma^\dagger\beta, \qquad \widehat{\Phi}_k = \widehat{\mathcal{P}}_{k+1}(C\,\Sigma)C\,\Sigma. \tag{4.31}$$

*where* $\mathcal{P}_k$ *and* $\widehat{\mathcal{P}}_{k+1}$ *are solution polynomials for GMRES and RRGMRES not exceeding degree* $k-1$ *and* $k$*, respectively. The filter matrices* $\Phi_k$ *and* $\widehat{\Phi}_k$ *are in general nondiagonal.*

PROOF. The relations (4.30)–(4.31) follow easily from describing the Krylov subspaces (4.11) and (4.26) by means of the SVD of $A$

$$\begin{aligned} \mathcal{K}_k(A,b) &= \operatorname{span}\left\{b, U\Sigma V^T b, \ldots, (U\Sigma V^T)^{k-1} b\right\} \\ \mathcal{K}_k(A,Ab) &= \operatorname{span}\left\{U\Sigma V^T b, (U\Sigma V^T)^2 b, \ldots, (U\Sigma V^T)^{k} b\right\}, \end{aligned}$$

and the "filter matrices" $\Phi_k$ and $\widehat{\Phi}_k$ are in general nondiagonal because $C = V^T U$ is in general nondiagonal. This shows directly that the singular values are "mixed", and that no simple filter in the SVD basis exists. □

Lemma 4.4 reveals that in general no method based on a Krylov subspace for the original system can be described as a spectral filtering method in the SVD sense if the coefficient matrix is nonsymmetric. Specifically, GMRES, RRGMRES, and even FOM in case of a positive definite (but nonsymmetric) coefficient matrix, cannot in general be expected to provide solutions similar to TSVD or Tikhonov solutions. Whether GMRES and its relatives can produce useful solutions to ill-posed problems is the subject of Section 4.3.

We noted earlier that if the coefficient matrix is normal then the convergence analyses of the Krylov subspace methods simplify. As we consider only real matrices $A \in \mathbb{R}^{n\times n}$, then normality can be defined as $A^T A = AA^T$. The following factorization holds for real normal matrices.

**Theorem 4.5** *Let $A \in \mathbb{R}^{n\times n}$ be a square normal matrix with real entries. Then the following factorization is defined*

$$A = QDQ^T,$$

*where $Q$ is orthogonal and $D$ has a diagonal structure with real $1\times 1$ or $2\times 2$ blocks. The $2\times 2$ blocks have the special structure*

$$D_i = \begin{pmatrix} a_i & b_i \\ -b_i & a_i \end{pmatrix}.$$

PROOF. See Horn and Johnson [46, §2.5] □

Note that the $2\times 2$ blocks in Theorem 4.5 are nothing but scaled Givens rotations

$$\begin{pmatrix} c_i & s_i \\ -s_i & c_i \end{pmatrix} = \frac{1}{\sigma_i}\begin{pmatrix} a_i & b_i \\ -b_i & a_i \end{pmatrix}, \quad \sigma_i = \sqrt{a_i^2 + b_i^2},$$

with $c_i = a_i/\sigma_i$ and $s_i = b_i/\sigma_i$. Now, for every $2\times 2$ block, let $D_i = \overline{D}_i \Sigma_i$ where $\overline{D}_i = D_i/\sigma_i$ and $\Sigma_i = \operatorname{diag}(\sigma_i, \sigma_i)$ holds the scaling factors. For every $1\times 1$ block let $\bar{d}_i = \operatorname{sign}(d_i)$ and $\sigma_i = |d_i|$. Collect all $\overline{D}_i$ and $\bar{d}_i$ in the orthogonal block diagonal matrix $\overline{D}$, and all scaling factors $\sigma_i$ and $\Sigma_i$ in the diagonal matrix $\Sigma$. Then an SVD of $A$ is given by $U\Sigma V^T$ with $U = V\overline{D}$ and it follows that $C = \overline{D}$ in (4.28). Therefore, for a real normal matrix $A$, the mixing of the singular components is limited to a

mixing of subspaces of dimension two. The factorization of Theorem 4.5 therefore in turn leads to a simpler description of the spectral properties of Krylov methods applied to the original system.

**Lemma 4.6** *Let $A \in \mathbb{R}^{n\times n}$ be real normal and let a Krylov method use the solution subspaces* (4.11) *or* (4.26)*, then* (4.30) *and* (4.31) *simplify to*

$$x^{(k)} = V\,\Phi_k\,\Sigma^\dagger\beta, \qquad \Phi_k = \mathcal{P}_k(\overline{D}\,\Sigma)\overline{D}\,\Sigma, \tag{4.32}$$

$$\hat{x}^{(k)} = V\,\widehat{\Phi}_k\,\Sigma^\dagger\beta, \qquad \widehat{\Phi}_k = \widehat{\mathcal{P}}_{k+1}(\overline{D}\,\Sigma)\overline{D}\,\Sigma. \tag{4.33}$$

*where $\overline{D}$ is the orthogonal block diagonal matrix as described above. The filter matrices $\Phi_k$ and $\widehat{\Phi}_k$ are also block diagonal with $1\times 1$ and $2\times 2$ blocks.*

Symmetric matrices trivially belong to the class of normal matrices. But in terms of specifying the solution subspaces in the SVD bases it improves the situation even further when $A$ is symmetric.

**Lemma 4.7** *Let $A \in \mathbb{R}^{n\times n}$ be symmetric, then Krylov methods using the solution subspaces* (4.11) *or* (4.26) *are spectral filtering methods in the SVD sense if the filter factors are allowed to be negative. The subspaces* (4.11) *and* (4.26) *can be described as*

$$V^T x^{(k)} \in \mathcal{K}_k\left(\Omega\Sigma, \Omega\beta\right), \qquad V^T \hat{x}^{(k)} \in \mathcal{K}_k\left(\Omega\Sigma, \Sigma\beta\right), \tag{4.34}$$

*where $\Omega \in \mathbb{R}^{n\times n}$ is a signature matrix. Equations* (4.30)–(4.31) *simplify to*

$$x^{(k)} = V\,\Phi_k\,\Sigma^\dagger\beta, \qquad \Phi_k = P_k(\Omega\,\Sigma)\Omega\,\Sigma, \tag{4.35}$$

$$\hat{x}^{(k)} = V\,\widehat{\Phi}_k\,\Sigma^\dagger\beta, \qquad \widehat{\Phi}_k = \widehat{\mathcal{P}}_{k+1}(\Omega\,\Sigma)\Omega\,\Sigma, \tag{4.36}$$

*where $\Phi_k = \mathcal{P}_k(\Omega\Sigma)\Omega\Sigma$ and $\widehat{\Phi}_k = \widehat{\mathcal{P}}_{k+1}(\Omega\Sigma)\Omega\Sigma$ are diagonal filter matrices.*

We observe from Lemma 4.7 that any Krylov method applied to the original system can be described as a spectral filtering method in the SVD basis whenever $A$ is symmetric. This includes MINRES and MR-II for possibly indefinite systems, and standard CG if $A$ is positive (semi-)definite. But even though these methods can be identified as special spectral filtering methods, the basis for the filter, i.e., the iteration matrix, is not the same as for CGLS and LSQR, that are based on the squared singular values $\Sigma^2$ arising from the normal equations formulation. That is, the convergence and the filtering properties are different.

### 4.2.3 Using Augmented Coefficient Matrix

Obviously, it is only possible to solve the system directly if the coefficient matrix is square. But what if $A$ is only given as a black-box function, and $A$ is nonsquare?

Calvetti et al. [8] proposed to perform zero-padding of the coefficient matrix and the right-hand side vector to turn an underdetermined system into a square system.

An overdetermined system can be made square in a similar manner by appending zero columns to $A$. Assume that $A \in \mathbb{R}^{m\times n}$ where $m < n$, and define

$$\widehat{A} = \begin{pmatrix} A \\ 0 \end{pmatrix} \quad \text{and} \quad \hat{b} = \begin{pmatrix} b \\ 0 \end{pmatrix},$$

such that $\widehat{A} \in \mathbb{R}^{n\times n}$ and $\hat{b} \in \mathbb{R}^n$. Now instead of computing $\min_x \|b - Ax\|_2$, we want to solve the equivalent problem $\min_x \|\hat{b} - \widehat{A}x\|_2$ to which GMRES-type algorithms can be applied.

It was shown in [8] by an example that GMRES and RRGMRES converge faster than CGLS, and moreover, to a solution with a smaller relative error. We will study this approach in the following as well.

## 4.3 Regularization Properties

So far, we have only seen examples on regularized solutions due to TSVD and Tikhonov regularization. Both of these methods can be described by means of a regularizing operator such that a regularized solution to the system $Ax = b$ is $x_\alpha = A_\alpha^\# b$ where $A_\alpha^\#$ is a regularized inverse of $A$ that leads to the regularized solution $x_\alpha$. Moreover, both methods produce solutions in a subspace mainly spanned by the first right singular vectors.

We continue along these lines, and get inspiration from the general definition of a regularization method in Definition 2.3. An iterative method applied to the system of linear equations (2.6) must somehow mimic a regularized operator when a suitable parameter choice method is devised.

In this setting, a suitable regularization parameter translates into a suitable number of iterations and therefore to the problem of devising a suitable stopping rule. Therefore, for a Krylov method to have regularization properties, it must be possible to determine the stopping index $k$ such that the solution $x^{(k)}$ is a regularized solution. Below, we define a stopping rule based on the discrepancy principle.

**Definition 4.8** (Stopping Rule based on Discrepancy Principle) Define the parameter $\nu > 1$ and assume that $\|b - b^\delta\| \leq \delta$. According to the discrepancy principle, the termination index is then the smallest index $k$ for which

$$\|b^\delta - Ax^{(k)}\| \leq \nu\delta,$$

where $x^{(k)}$ is the $k$th solution iterate of a Krylov subspace method.

Often, the only measures we have around are the residual norms and the solution (semi-)norms. For practical problems we definitely do not have the true solution, and computing the relative error $\varepsilon_2$ from (2.17) with respect to the true solution can be done only for test problems to verify the regularization properties and the stopping rules. Therefore, some measure involving the residual seems to be the way to go when implementing stopping rules.

We now look at the potentials of using the two Krylov subspaces $\mathcal{K}_k(A, b)$ and $\mathcal{K}_k(A, Ab)$ based on the original system, and $\mathcal{K}_k(A^TA, A^Tb)$ based on the normal equations as subspaces for regularized solutions. There are several aspects to consider when investigating the regularization properties of Krylov methods.

**The solution subspace.** Is a suitable solution contained in the generated Krylov subspace? That is, is it possible to find an $x^{(k)} \in \mathcal{K}_k$, where $\mathcal{K}_k$ is the Krylov subspace of the iterative method, such that the relative error $\varepsilon_2$ (or some other quality measure) is small?

**The minimization or orthogonality properties.** Does the Krylov method approximate a suitable solution from a given Krylov subspace? That is, if a given solution subspace $\mathcal{K}_k$ does contain a solution $x^{(k)} \in \mathcal{K}_k$ such that the relative error $\varepsilon_2$ is small, will the Krylov method find it?

**Convergence.** Given a series of Krylov subspaces, how does a series of approximated solutions behave and which one is the best? Is it possible to devise a stopping rule based on the number of iterations?

It is obvious from the previous section that any combination of solution subspace and minimization property can be constructed. Furthermore, the symmetry of the coefficient matrix is important when using methods based on the Lanczos tridiagonalization scheme. A short summary of the properties is seen in Table 4.1.

Traditionally, the most studied and most used algorithms are CGLS and LSQR, followed by MINRES, and from the Table 4.1, we clearly see why this is the case. These two fields of the table indicate algorithms that minimize the residual of the original system $r^{(k)} = b - Ax^{(k)}$ in each iteration, and furthermore, produce solutions in subspaces that are connected to a spectral filtering in the SVD domain. Also the MINRES variant MR-II falls into this category. Therefore, we take a closer look at these algorithms.

The GMRES method is also mentioned several places in the literature, and its regularizing properties have been slightly studied [11]. It is still a minimum-residual method, but for nonsymmetric coefficient matrices, the algorithm does not perform spectral filtering in terms of the SVD. We do not here look at either CG for symmetric and positive (semi-)definite systems, or MINRES applied to the normal equations. The first is studied theoretically in e.g., [31] and rarely applied to real-world problems, and the latter does not minimize the residual norm of the original system, but rather the residual of the normal equations. For the same reason, FOM applied to the original (possibly nonsymmetric) system, Craig's method, and minimum residual methods applied to $AA^Ty = b$, $x = A^Ty$ are also avoided here.

### 4.3.1 The Krylov Solution Subspaces

We look more specifically at the methods that minimize the residual in each iteration, i.e., CGLS/LSQR, GMRES/MINRES, and RRGMRES/MR-II. First we note that

| | OR Approach | MR Approach |
|---|---|---|
| $Ax = b,\ A \neq A^T$ | **If $A$ indefinite:**<br>No OR-approximation is guaranteed<br>**If $A$ positive (semi-)definite:** FOM<br><br>**Minimizing property:**<br>$\min \|x - x^*\|_A,\ \text{s.t}\ x \in \mathcal{K}(A, b)$<br>*NOT spectral filtering* | GMRES (RRGMRES)<br><br>**Minimizing property:**<br>$\min \|r\|_2,$<br>$\text{s.t}\ x \in \mathcal{K}(A, b),\ (\mathcal{K}(A, Ab))$<br>*NOT spectral filtering* |
| $Ax = b,\ A = A^T$ | **If $A$ indefinite:**<br>No OR-approximation is guaranteed<br>**If $A$ positive (semi-)definite:**CG<br><br>**Minimizing property:**<br>$\min \|x - x^*\|_A,\ \text{s.t}\ x \in \mathcal{K}(A, b)$<br>*Spectral filtering* | MINRES (MR-II)<br><br>**Minimizing property:**<br>$\min \|r\|_2,$<br>$\text{s.t}\ x \in \mathcal{K}(A, b),\ (\mathcal{K}(A, Ab))$<br>*Spectral filtering* |
| $A^TAx = A^Tb$ | CGLS/LSQR<br><br>**Minimizing property:**<br>$\min \|r\|_2,\ \text{s.t}\ x \in \mathcal{K}(A^TA, A^Tb)$<br>*Spectral filtering* | MINRES (MR-II)<br><br>**Minimizing property:**<br>$\min \|A^Tr\|_2,$<br>$\text{s.t}\ x \in \mathcal{K}(A^TA, A^Tb),\ \big(\mathcal{K}(A, A^TAA^Tb)\big)$<br>*Spectral filtering* |
| $AA^Ty = b,\ x = A^Ty$ | CGNE (Craig)<br><br>**Minimizing property:** (if defined)<br>$\min \|x^* - x\|_2,\ \text{s.t}\ x \in \mathcal{K}(A^TA, A^Tb)$<br>*Spectral filtering* | MINRES (MR-II)<br><br>**Minimizing property:**<br>$\min \|b - AA^Ty\|_2,$<br>$\text{s.t}\ y \in \mathcal{K}(AA^T, b),\ \big(\mathcal{K}(AA^T, AA^Tb)\big)$<br>*Spectral filtering* |

Table 4.1: Algorithms that compute OR and MR approximations to $Ax = b$, $A^TAx = A^Tb$, and $AA^Ty = b$, $x = A^Ty$. The residual of the original system is defined as $r = b - Ax$.

discrete ill-posed problems in practise due to effects of the finite precision representation, can be considered as rank-deficient problems. That is the singular values for the coefficient matrix hit the machine precision at some index $r < n$, and we will define $r$ as the numerical rank. A discrete problem that still appears as truly rank-deficient will have a large gap between the large singular values and the small at machine precision. On the other hand, both truly ill-posed problems and rank-deficient problems where the smallest nonzero singular value is smaller than the machine precision will appear identically in the discrete setting. Consider now Table 4.2. This table indicates that only CGLS and LSQR always produce solutions in $\mathcal{R}(A^T)$ similar to TSVD and Tikhonov regularization, regardless of the structure of $A$. In fact, Hanke [32] observed that CGLS and LSQR are very often superior to TSVD. MR-II produces solutions in a subspace of $\mathcal{R}(A^T)$ whenever $A$ is symmetric, and MINRES whenever $A$ is symmetric and the system is consistent. The RRGMRES method can also produce solutions in $\mathcal{R}(A^T)$, but this requires that $A$ is *range-symmetric*,

| Method | Krylov subspace | Subspace | System |
|---|---|---|---|
| CGLS: | $\mathcal{K}_k\left(A^TA, A^Tb\right)$ | $\subseteq \mathcal{R}(A^T)$ | any |
| MR-II: | $\mathcal{K}_k\left(A, Ab\right) = \mathcal{K}_k\left(A^T, A^Tb\right)$ | $\subseteq \mathcal{R}(A^T)$ | symmetric |
| MINRES: | $\mathcal{K}_k\left(A, b\right) = \mathcal{K}_k\left(A^T, b\right)$ | $\subseteq \mathcal{R}(A^T)$ | symmetric/consistent |
| RRGMRES: | $\mathcal{K}_k\left(A, Ab\right)$ | $\subseteq \mathcal{R}(A^T)$ | range-symmetric |
| GMRES: | $\mathcal{K}_k\left(A, b\right)$ | $\subseteq \mathcal{R}(A^T)$ | range-symmetric/consistent |
| GMRES: | $\mathcal{K}_k\left(A, b\right)$ | $\subseteq \mathcal{R}(A)$ | consistent |
| RRGMRES: | $\mathcal{K}_k\left(A, b\right)$ | $\subseteq \mathcal{R}(A)$ | any |
| MINRES: | $\mathcal{K}_k\left(A, b\right) = \mathcal{K}_k\left(A^T, b\right)$ | $\subseteq \mathcal{R}(A^T) + b$ | symmetric/inconsistent |
| GMRES: | $\mathcal{K}_k\left(A, b\right)$ | $\subseteq \mathcal{R}(A) + b$ | inconsistent |

Table 4.2: Illustration of solution subspaces for the various methods. Here $\mathcal{R}(A) = (\, u_1, \ldots, u_r \,)$, $\mathcal{R}(A^T) = (\, v_1, \ldots, v_r \,)$ where $r < n$ is the numerical rank of $A$.

i.e., that $\mathcal{R}(A) = \mathcal{R}(A^T)$. Finally, for GMRES to produce solutions in $\mathcal{R}(A^T)$ it is additionally required that the system is consistent.

Unfortunately, discrete ill-posed problems in practise behave as rank-deficient problems due to finite-arithemetic effects and the decaying singular values. We can therefore define a numerical rank connected to the machine precision, and the systems are often inconsistent, either because of measurement noise or due to discretization errors. Furthermore, because of the noise, we are often not interested in constructing solutions in the entire $\mathcal{R}(A^T)$ because some of the components will possibly be severely affected by the noise. Thus Table 4.2 only indicates which methods we can hope will provide suitable regularized solutions similar to TSVD and Tikhonov solutions.

For symmetric matrices, Eqs. (4.25) and (4.34), together with the discrete Picard condition, imply that all the Krylov vectors have elements which, on average, decay for increasing index $i$. However, due to the different powers of $\Sigma$, the damping imposed by the multiplication with the singular values are different for these methods. For example, the $k$th CGLS/LSQR Krylov vector is equal to the $2k$th Krylov vector of MINRES and the $(2k-1)$st Krylov vector of MR-II. Moreover, the vector $\beta = U^Tb$ appears undamped in the MINRES basis, while in the CGLS and MR-II bases it is always damped by $\Sigma$. In the presence of noise, the fact that $\beta$ appears undamped in the MINRES Krylov subspace can have a dramatic impact, as we illustrate below.

For nonsymmetric matrices the behavior of CGLS/LSQR is identical to the symmetric case. On the other hand, the Krylov vectors for GMRES and RRGMRES from (4.29) are different; even if the discrete Picard condition is fulfilled, we cannot be sure that the coefficients $|v_i^Tb|$ decay, on average, for increasing index $i$. Furthermore, due to the presence of the non-diagonal matrix $C\Sigma$, no structured damping of these coefficients is obtained because the SVD components are "mixed." This means that GMRES and RRGMRES, in general, cannot be assumed to produce a solution subspace that resembles the one spanned by the first right singular vectors. Consider the following examples.

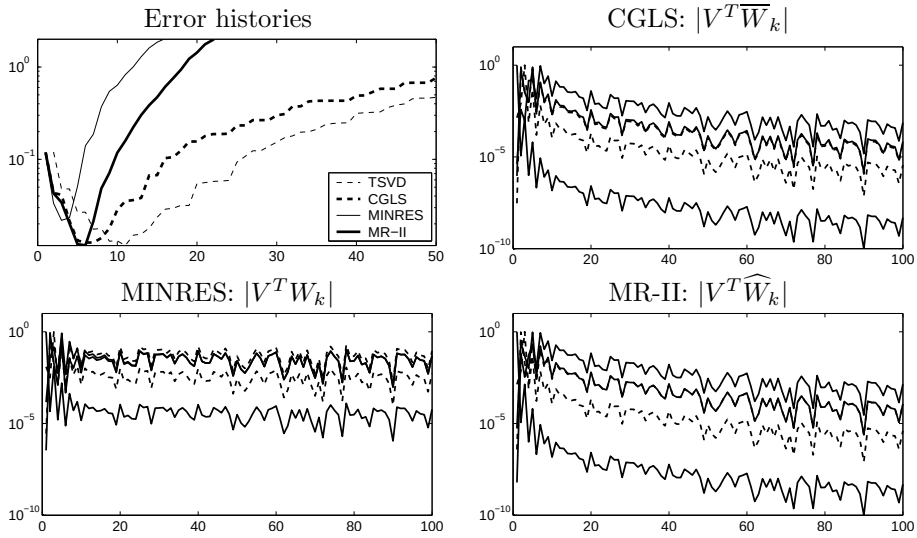



Figure 4.1: Symmetric deriv2 test problem. Top left: the relative errors $\varepsilon_2(\bar{x}^{(k)})$, $\varepsilon_2(x^{(k)})$ and $\varepsilon_2(\hat{x}^{(k)})$ for CGLS, MINRES and MR-II, and $\varepsilon_2(x_k)$ for TSVD. Remaining plots: the first five orthonormal Krylov vectors of the CGLS, MINRES and MR-II subspaces in the SVD basis.

**Example 4.1** *We use the symmetric problem deriv2 from MOORe Tools [49] with a coefficient matrix of size* $100 \times 100$*, and we use the third implemented solution. We add white Gaussian noise* $e$ *to the right-hand side* $b$ *such that* $b^\delta = b + e$ *with* $\|e\|_2/\|b\|_2 = 5 \cdot 10^{-4}$*. Figure 4.1 shows the relative errors for a series of CGLS, MINRES and MR-II iterates, as well as the relative errors of similar TSVD solutions. Note that MINRES does not reduce the relative error as much as the other methods due to the noise component in the initial Krylov vector. Also, observe that MR-II and CGLS reduce the relative error to about the same level, 0.0117 for MR-II and 0.0125 for CGLS, in 5–6 iterations. This makes MR-II favorable for this problem, because the number of matrix-vector multiplications is halved compared to CGLS. The best TSVD solution includes 11 SVD components. This indicates that the CGLS and MR-II solution subspaces are superior compared to the TSVD solution subspace of equal dimensions, and therefore supports the results from [32].*

*Figure 4.1 also shows the first five orthonormal Krylov vectors expressed in terms of the right singular vectors* $v_i$*, i.e., the first five columns of* $|V^T\overline{W}_k|$*,* $|V^T W_k|$ *and* $|V^T\widehat{W}_k|$ *for CGLS, MINRES and MR-II, respectively. We see that the CGLS and MR-II vectors are mainly spanned by the first right singular vectors as expected, and that the contribution from the latter singular vectors is damped. We also see that the contribution from the latter right singular vectors is much more pronounced for*

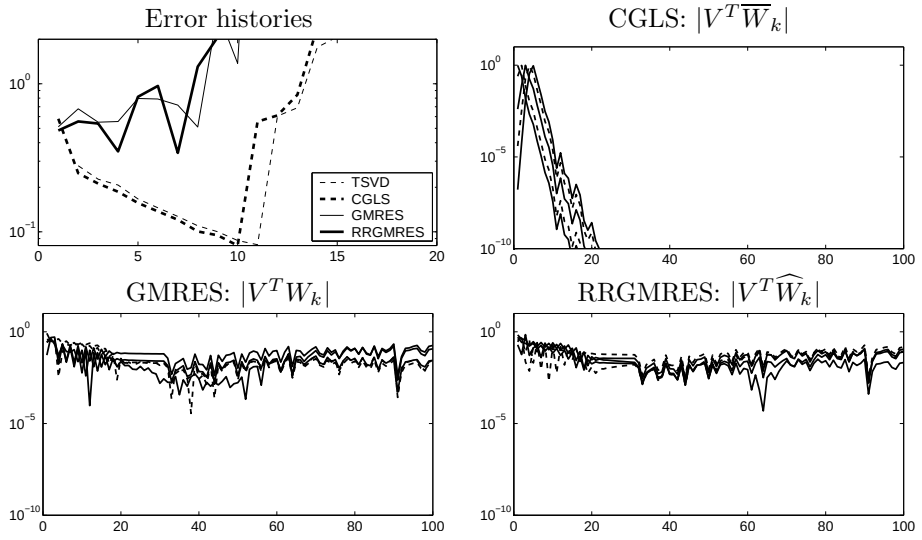



Figure 4.2: Nonsymmetric ilaplace test problem. Top left: the relative errors $\varepsilon_2(\bar{x}^{(k)})$, $\varepsilon_2(x^{(k)})$ and $\varepsilon_2(\hat{x}^{(k)})$ for CGLS, GMRES and RRGMRES, and $\varepsilon_2(x_k)$ for TSVD. Remaining plots: the first five orthonormal Krylov vectors of the CGLS, GMRES and RRGMRES subspaces in the SVD basis.

*MINRES due to the direct inclusion of the noise.*

**Example 4.2** *Consider the nonsymmetric problem* ilaplace *from MOORe Tools [49] with a coefficient matrix of size* $100 \times 100$ *and additive white Gaussian noise e with* $\|e\|_2/\|b\|_2 = 5 \cdot 10^{-4}$. *Figure 4.2 shows plots similar to those for the symmetric case. Only TSVD and CGLS are able to reduce the relative error considerably; neither GMRES nor RRGMRES produce iterates with small relative errors, and the "convergence" is erratic.*

*We note that the CGLS Krylov vectors behave similar to the symmetric case, i.e., the components that correspond to small singular values are effectively damped. Furthermore, we see that the GMRES and RRGMRES Krylov vectors do not exhibit any particular damping. In fact, all Krylov subspace vectors contain significant components along all right singular vectors – including those that correspond to small singular values. Therefore the GMRES and RRGMRES iterates are composed not only of the first right singular vectors, but also include significant components in the direction of the last right singular vectors.*

The regularization properties of CGLS and LSQR have been studied extensively both theoretically and in a more practical framework [22, 31, 33, 34, 73, 92]. It is a well-known result that the stopping rule from Definition 4.8 can be used with

CGLS and LSQR, and that the solution polynomials can be considered as regularized operators in correspondence with Definition 2.3. In the following, we will investigate the behavior of MINRES, GMRES, MR-II and RRGMRES when applied to discrete ill-posed problems.

### 4.3.2 Symmetric $A$ – MINRES and MR-II

Lemma 4.7 characterizes MINRES and MR-II as spectral filtering methods in the SVD of the symmetric matrix $A$ if the filter factors are allowed to be negative. But it is also clear that both MINRES and MR-II can be described through polynomials in the eigenvalues of $A$. Note that $A = U\Sigma V^T = V\Omega\Sigma V^T$ is an eigenvalue decomposition of $A$ where $\Lambda = \Omega\Sigma = \mathrm{diag}(\,\lambda_1, \ldots, \lambda_n\,)$ are the eigenvalues, sorted in (absolute) nonincreasing order. Moreover, the right singular basis $V$ is an eigenvector basis. We now write the solution iterates and the corresponding residuals for MINRES and MR-II as

$$x^{(k)} = V\mathcal{P}_k(\Lambda)V^T b \qquad \text{and} \qquad r^{(k)} = V\mathcal{Q}_k(\Lambda)V^T b \tag{4.37}$$

$$\hat{x}^{(k)} = V\widehat{\mathcal{P}}_{k+1}(\Lambda)V^T b \qquad \text{and} \qquad \hat{r}^{(k)} = V\widehat{\mathcal{Q}}_{k+1}(\Lambda)V^T b, \tag{4.38}$$

where $\mathcal{P}_k$, $\widehat{\mathcal{P}}_{k+1}$, $\mathcal{Q}_k$, and $\widehat{\mathcal{Q}}_{k+1}$ are the solution and residual polynomials for the two methods. The signs in $\Omega$ indicate the definiteness of $A$, and it is well-known, see e.g., [63], that the convergence of MINRES is affected by the definiteness of the coefficient matrix.

Let us study the residual polynomials and the minimization properties of MINRES and MR-II. From (4.6) we know that any residual polynomial $\mathcal{Q}_k$ must satisfy $\mathcal{Q}_k(0) = 1$, i.e, $\mathcal{Q}_k(0) = \widehat{\mathcal{Q}}_{k+1}(0) = 1$. Furthermore, we note from (4.34) and (4.36) that the constant term of $\widehat{\mathcal{P}}_{k+1}$ is zero; and it follows that $\widehat{\mathcal{Q}}'_{k+1}(0) = 0$. The residual norms can now be written as

$$\|b - A\,x^{(k)}\|_2 = \|\mathcal{Q}_k(\Lambda)V^T b\|_2 \quad \text{and} \quad \|b - A\,\hat{x}^{(k)}\|_2 = \|\widehat{\mathcal{Q}}_{k+1}(\Lambda)V^T b\|_2,$$

for MINRES and MR-II, respectively.

If the discrete Picard condition is satisfied then the elements of the vector $V^T b$ decay overall faster than the singular values, i.e., faster than the absolute eigenvalues. Therefore the first components of $V^T b$ will in general be larger than the latter. To minimize the residual norm, it will be beneficial for the residual polynomials to be small for the large absolute eigenvalues, i.e., for an indefinite system $\mathcal{Q}_k$ and $\widehat{\mathcal{Q}}_{k+1}$ should be small for the extreme eigenvalues, both positive and negative, while maintaining a value of one at the origin. On the other hand, if a specific right-hand side component $v_j^T b$ is small, then the polynomial need not necessarily be small at the corresponding eigenvalue $\lambda_j$ because this component does not strongly affect the residual norm.

Several authors investigate the problem of approximating zero on a set of eigenvalues with a residual polynomial, see e.g., Fischer [23] and Greenbaum [28, §3.1]. A quite interesting result appears in [23, Theorem 6.9.9]. Here it is shown that if $A$ has

a symmetric expansion into orthonormal eigenvectors, i.e., if the eigenvalues for an orthonormal eigenvector basis are distributed symmetrically in two intervals around zero, then $x^{(2n+1)} = x^{(2n)} = \bar{x}^{(n)}$ where $x^{(2n+1)}$ and $x^{(2n)}$ are the MINRES iterates after $2n+1$ and $2n$ iterations, respectively, and $\bar{x}^{(n)}$ is the CGLS iterate after only $n$ iterations. That is, the degree of the MINRES solution polynomial needs to be twice as high as the degree of the CGLS solution polynomial to generate an equivalent solution. Despite the fact that two matrix-vector multiplications are needed in each CGLS iteration, and only one is needed in each MINRES iteration, CGLS may be favorable because the additional work is not twice as expensive.

The regularization properties of MINRES and MR-II have been studied somewhat in the literature [31, 56, 57, 58]. Regardless of the definiteness of the coefficient matrix it is shown [56, Theorem 3.1] that the MINRES residual cannot be reduced to a value below the norm of the error in the right-hand side without harming the solution iterate. This indicates that a stopping rule based on the discrepancy principal indeed works for MINRES. Similar conclusions can be drawn for both MINRES and MR-II from the more theoretical results due to Hanke [31].

#### 4.3.2.1 Numerical Examples

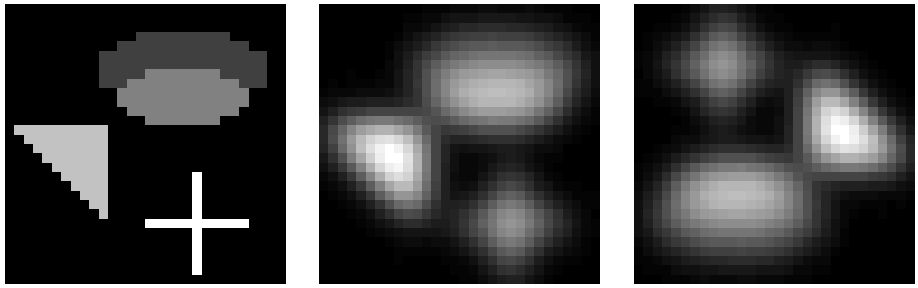
Several of the examples in the literature are constructed in a simplified framework to illustrate specific parts of the theory. In the following examples, we apply MINRES, MR-II and LSQR, to variants of a more realistic, though still constructed, image reconstruction problem. The aim is to illustrate that the convergence of MINRES and MR-II are very affected by even different formulations of very similar problems, but that they nevertheless have a regularizing effect.

**Example 4.3** *Let $X_{\text{exact}} \in \mathbb{R}^{30\times 30}$ be the sharp image seen in Fig. 4.3 (left), and let the PSF matrix $A \in \mathbb{R}^{900\times 900}$ be a discretization of a two-dimensional Gaussian kernel* (3.9) *with zero boundary conditions, and $\sigma_s = \sigma_t = 2$. Let* vec *and* $\text{vec}^{-1}$ *be defined as in Definitions 3.3–3.4 and let $x_{\text{exact}} = \text{vec}\,(X_{\text{exact}})$. Let $e \in \mathbb{R}^{900}$ be a vector of white Gaussian noise, scaled such that $\|e\|_2/\|Ax_{\text{exact}}\|_2 = 10^{-2}$. The blurred right-hand side is given by $b^\delta = Ax_{\text{exact}} + e$, and the noisy image $B^\delta = \text{vec}^{-1}\left(b^\delta, 30, 30\right)$ is seen in Fig. 4.3 (middle). The coefficient matrix $A$ is both symmetric, $A = A^T$, and symmetric with respect to the main skew diagonal, i.e., if $P \in \mathbb{R}^{900\times 900}$ is the orthogonal reversal matrix, then also $PA = (PA)^T$. The 2-norm is not affected by an orthogonal transformation, and it follows that the two problems*

$$\min_x \|b^\delta - A\,x\|_2 \qquad \text{and} \qquad \min_x \|Pb^\delta - PA\,x\|_2, \tag{4.39}$$

*are identical in the least squares sense. The permuted right-hand side $Pb^\delta$ corresponds to a $180^\circ$ rotation of the original right-hand side and $\text{vec}^{-1}\left(Pb^\delta, 30, 30\right)$ is shown in Fig. 4.3 (right).*

*Because both $A$ and $PA$ are symmetric then MINRES and MR-II, in addition to LSQR, can be applied both problems* (4.39)*, and we perform 100 iterations without reorthogonalization of the Krylov vectors. Fig. 4.4 reports the reduction of the*



$X_{\text{exact}}$ $B^{\delta}$ $\text{vec}^{-1}\left(Pb^{\delta},\, 30,\, 30\right)$

Figure 4.3: True and blurred images for MINRES example.

| Problem | LSQR | MINRES | MR-II |
|---|---|---|---|
| $\min_x \|b^{\delta} - Ax\|_2$ | 0.4680 (59) | 0.4916 (6) | 0.4678 (20) |
| $\min_x \|Pb^{\delta} - PAx\|_2$ | 0.4680 (59) | 0.4682 (81) | 0.4681 (79) |

Table 4.3: Optimal relative errors and number of iterates for LSQR, MINRES and MR-II when applied to the two problems (4.39).

*residuals for the three methods and the two versions of the problem. Obviously, the convergence of LSQR is the same for the two problems, whereas the convergence of MINRES and MR-II are* faster *in the first case, and* slower *in the second case. We compute the relative errors compared to the true solution $x_{\text{exact}}$ and the Table 4.3 shows the optimal solution iterates and the corresponding relative errors. Note that for the first formulation of the problem both MINRES and MR-II converge to their optimal solutions considerably faster than LSQR; MINRES to a solution iterate with a slightly higher relative error, and MR-II to an iterate comparable to the LSQR solution iterate. For the second formulation, both MINRES and MR-II require more iterations than LSQR, and now both converge to solutions comparable to the LSQR solution. Note that MINRES and MR-II are still favorable in terms of computational work because only one matrix-vector multiplication is needed in each iteration compared to two for LSQR.*

At a first glance, Example 4.3 shows two formulations of the same least squares problem. Yet very different convergence histories are observed for both MINRES and MR-II. Even though the minimization properties of the three Krylov methods are the same (they all minimize the 2-norm of the residual), the solution subspaces are different. The LSQR solution subspaces are identical for the two formulations in (4.39) whereas the MINRES and MR-II solution subspaces differ. We have:

$$\text{LSQR} : \mathcal{K}_k(A^TA,\, A^Tb) = \mathcal{K}_k(A^TP^TPA,\, A^TP^TPb)$$

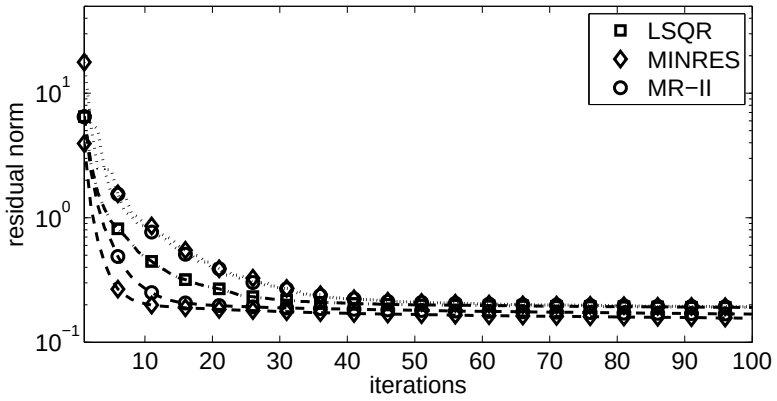



Figure 4.4: Convergence of residual for MINRES, MR-II, and LSQR applied to $\|b^\delta - Ax\|_2$ (dashed lines), and $\|Pb^\delta - PAx\|_2$ (dotted lines).

$$\text{MINRES} : \mathcal{K}_k(A,\, b) \neq \mathcal{K}_k(PA,\, Pb),$$

$$\text{MR} - \text{II} : \mathcal{K}_k(A,\, Ab) \neq \mathcal{K}_k(PA,\, PAPb) = \mathcal{K}_k(PA, Ab)\ .$$

We also note that $A$ is symmetric and positive definite, whereas $PA$ is indefinite. Moreover, the right-hand side $Pb^\delta$ has significant components corresponding to both positive and negative eigenvalues of $PA$.

If we formulate the LSQR residual in terms of the residual polynomial $\overline{\mathcal{Q}}_k(\Sigma^2)$ we note that due to the formulation through the normal equations, this does not depend on the definiteness of $A$. The effect of the polynomial is the same as for MINRES and MR-II: "kill" all significant right-hand side components. But this polynomial has the big advantage that all the squared singular values are positive.

To study the residual polynomials for the three methods, we illustrate how the polynomial coefficients for the MINRES residual polynomial $\mathcal{Q}_k$ can be obtained. First we define the matrix $L \in \mathbb{R}^{n\times k}$ as $L = (\lambda^1,\, \lambda^2,\, \ldots,\, \lambda^k)$ where $\lambda^i = \text{diag}(\Lambda^i)$ is the vector of eigenvalues to the $i$th power. Furthermore, $\text{diag}(I)$ denotes the vector of all ones.

$$r^{(k)} = V\mathcal{Q}_k(\Lambda)V^T b = V\left(I - \text{diag}\left(L\,\gamma^{(k)}\right)\right)V^T \tag{4.40}$$

$$= V\text{diag}(V^T b)\left(\text{diag}(I) - L\,\gamma^{(k)}\right). \tag{4.41}$$

To obtain (4.41) from (4.40) we use that for two vectors $a,\, b \in \mathbb{R}^n$ we have that $\text{diag}(a)b = \text{diag}(b)a$. The vector of coefficients for the MINRES solution polynomial after $k$ iterations is thus given by

$$\gamma^{(k)} = -\left(V\text{diag}(V^T b)L\right)^\dagger\left(r^{(k)} - V\text{diag}(V^T b)\right). \tag{4.42}$$

Because $\mathcal{Q}_k = I - \Lambda\mathcal{P}_k(\Lambda)$, the vector of coefficients for the corresponding residual polynomial is therefore given as $\varrho = (\,1\,,\, -\gamma^{(k)T}\,)^T$. The polynomial coefficients for MR-II and LSQR can be defined similarly.

In the next example, we look at some of the residual polynomials for the three methods when applied to the problems in Example 4.3.

**Example 4.4** *We continue Example 4.3, and show in Fig. 4.5 the distribution of the eigenvalues or squared singular values together with the size of the right-hand side coefficients corresponding to those. The right-hand side coefficients are normalized such that the largest is one. For MINRES and MR-II, we show the first four residual polynomials in terms of the eigenvalues, and for LSQR we show the four first residual polynomials in terms of the squared singular values. The situation is shown for both*

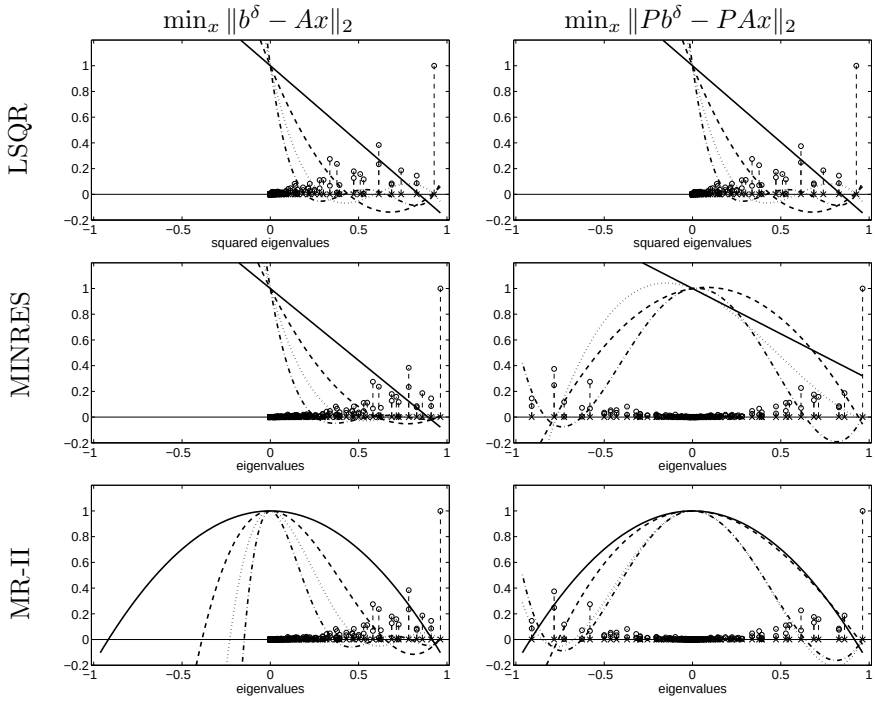



Figure 4.5: First four residual polynomials (solid, dashed, dotted, and dash-dotted) for LSQR, MINRES, and MR-II. Plots in the left side correspond to the original problem $\|b^\delta - Ax\|_2$, and plots in the right side correspond to the permuted problem $\|Pb^\delta - PAx\|_2$. The crosses indicate the eigenvalues (MINRES and MR-II) and squared singular values (LSQR) and the vertical lines indicate the size of the right-hand side component for a given eigenvalue/squared singular value.

*formulations of the problem* (4.39).

*Note how the MINRES and MR-II polynomials need to "kill" components corresponding to both positive and negative eigenvalues for the permuted problem. The situation for LSQR, on the other hand, is identical for the two formulations. It is interesting to see that the right-hand side component is large for the largest positive eigenvalue, and therefore all MINRES and MR-II polynomials are small around this eigenvalue. On the other hand, the components corresponding to the largest negative eigenvalue is not as large as the components corresponding to the second largest negative eigenvalue. Therefore, the polynomials are generally smaller for the second largest negative eigenvalue than for the largest.*

For this very visual image deblurring example, there is another way to justify the slower convergence of the second formulation of the problem. Furthermore, we can argue why MINRES applied to the second problem can produce iterates with smaller relative errors than when applied to the first problem.

Obviously, $A$ performs a blurring and $P$ performs a 180° rotation. Therefore, the first Krylov vector in the MINRES solution subspace $Pb^\delta = PAx_{\text{exact}} + e$ is mainly a 180° rotation of the blurred true solution. Comparing the true solution $X_{\text{exact}}$ and the blurred permuted image $\text{vec}^{-1}\left(Pb^\delta, 30, 30\right)$ from Fig. 4.3, we see that there is only a small pixel-wise correspondence between the two images. Therefore, this Krylov vector will only contribute very little to the solution, and the noise present in the first undamped Krylov vector will not be strongly present in the solution iterates. If we look at a the three first Krylov vectors

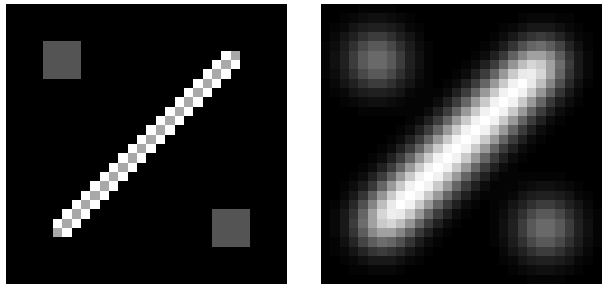
$$PAx_{\text{exact}}, \quad PAPAx_{\text{exact}} = A^2x_{\text{exact}}, \quad PAPAPAx_{\text{exact}} = PA^3x_{\text{exact}}, \quad \ldots$$

we see that every second Krylov vector is permuted. In comparison, all the Krylov vectors for the original formulation of the problem do correspond pixel-wise to the true solution; also the first noisy vector in the MINRES basis. This indeed indicates that the convergence of the solution iterates based on the permuted problem will be slower than for the original problem, and that MINRES iterates based on the original problem will more likely include noise.

Now let us look at a slightly modified problem where the right-hand side is invariant under a 180° rotation.

**Example 4.5** *We use the same coefficient matrices as in Example 4.3, but create a true solution* $x'_{\text{exact}} = \text{vec}\,(X'_{\text{exact}})$ *which is invariant under a* 180° *rotation, i.e.,* $x'_{\text{exact}} = Px'_{\text{exact}}$. *Therefore, the true right-hand side is also invariant under rotation;* $Pb' = PAx'_{\text{exact}} = PAPx'_{\text{exact}} = Ax'_{\text{exact}} = b'$. *The true and the blurred images are seen in Fig. 4.6. Note that we consider the noise-free right-hand side.*

*In this case, the residual for MINRES and MR-II converge faster than the residual for LSQR, both when applied to* $\|b' - Ax'\|_2$ *and* $\|Pb' - PAx'\|_2$ *as seen in Fig. 4.7. This behavior is observed because the right-hand side components that correspond to negative eigenvalues are practically zero. Even though the coefficient matrix* $PA$ *is indefinite, the residual polynomials need mainly to deal with the positive eigenvalues.*



$\mathrm{vec}^{-1}\,(x'_{\mathrm{exact}},\,30,\,30)$ $\mathrm{vec}^{-1}\,(b',\,30,\,30)$

Figure 4.6: True and blurred images for rotationally invariant case.

*This situation is illustrated in Fig. 4.8. Note how the residual polynomials are allowed to grow even when negative eigenvalues exist. The fast convergence can also be explained by looking at the images. Obviously, there is a pixel-wise correspondence between the $X'_{\mathrm{exact}}$ and all the Krylov vectors seen as images. Therefore all Krylov vectors contribute to the solution.*

Example 4.3–4.5 illustrate how the convergence of MINRES and MR-II depend on the behavior of the residual polynomials. They also illustrate that the residual polynomials depend both on the location of the eigenvalues of the system matrix, and on the right-hand side components expressed in the eigenbasis. In Example 4.5, the noise-free right-hand side was used to illustrate the behavior of the residual polynomials when the coefficients corresponding to the largest negative eigenvalues were effectively zero. For a noisy right-hand side, even tiny noise component corresponding to the negative eigenvalues will influence the residual polynomials and therefore the convergence.

A final example illustrates how the stopping rule from Definition 4.8 based on the discrepancy principle makes sense for MINRES and MR-II regardless of the formulation of the problem.

**Example 4.6** *We consider all the four problems from the examples above, and consider only noisy right-hand sides*

$$\min_x \|b^\delta - Ax\|_2, \qquad \min_x \|Pb^\delta - PAx\|_2 \tag{4.43}$$

$$\min_{x'} \|b'^\delta - Ax'\|_2, \qquad \min_{x'} \|Pb'^\delta - PAx'\|_2\ . \tag{4.44}$$

*A noise vector e is generated and a number of different scalings are used such that $\|e\|_2/\|b\|_2 \in \{10^{-5}; 10^0\}$ and $\|e\|_2/\|b'\|_2 \in \{10^{-5}; 10^0\}$. We define $\nu = 1.1$ for the stopping rule in Definition 4.8 and perform a maximum of 2000 iterations of LSQR, MINRES and MR-II without reorthogonalization applied to all four problems and for the variety of noise levels. For each method and each noise level we locate the iterate*

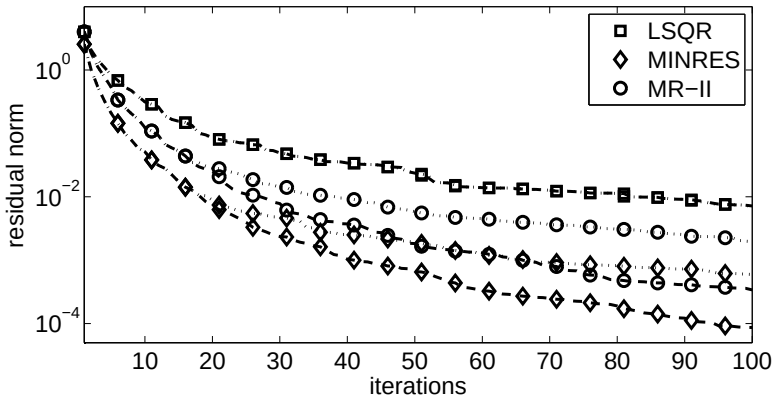



Figure 4.7: Residuals of LSQR, MINRES and MR-II when applied to $\|b' - Ax'\|_2$ (dashed lines), and $\|Pb' - PAx'\|_2$ (dotted lines).

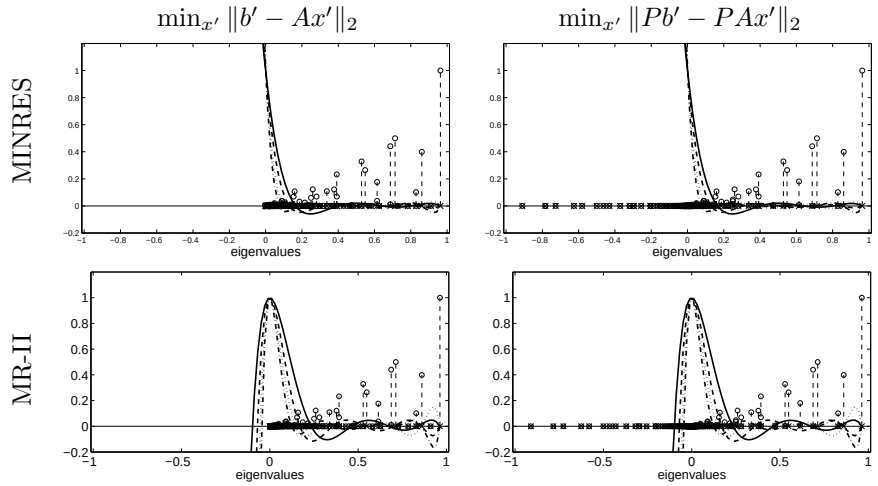



Figure 4.8: Residual polynomials 5–8 for MINRES and MR-II. Plots in the left side correspond to the original problem $\|b' - Ax'\|_2$, and plots in the right side correspond to the permuted problem $\|Pb' - PAx'\|_2$. As in Fig. 4.5, the crosses indicate the eigenvalues, and the vertical lines indicate the size of the right-hand side component for a given eigenvalue.

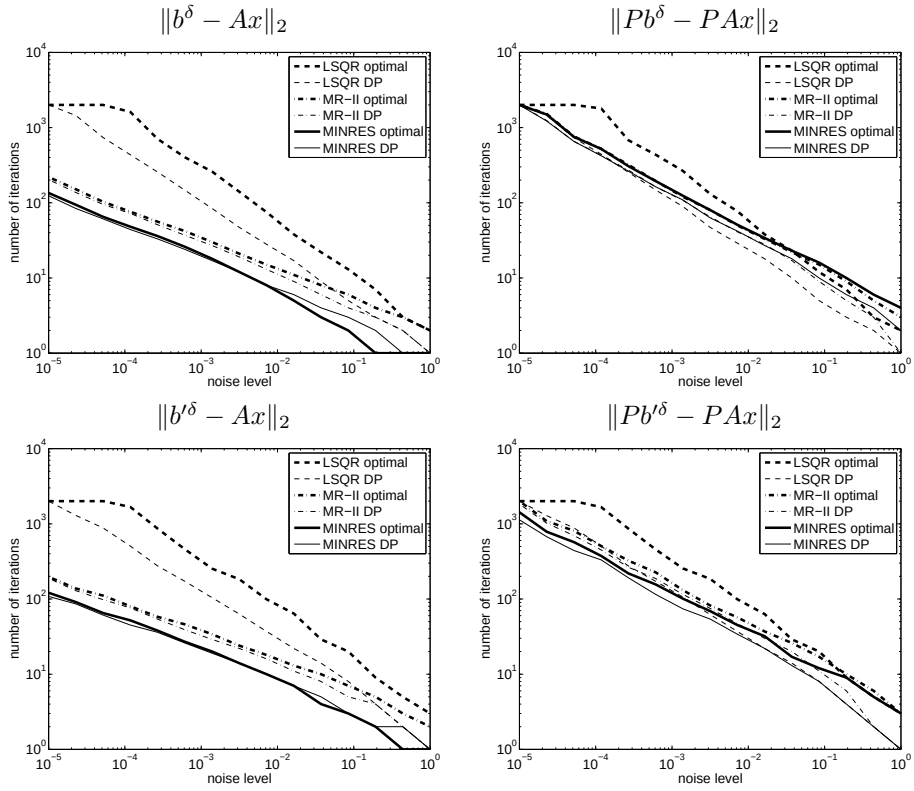



Figure 4.9: Correspondence between amount of noise in the right-hand side and the optimal number of iterates of LSQR, MINRES and MR-II (thick lines). Also shown are the iterates corresponding to the stopping rule from Definition 4.8 with $\nu = 1.1$ (thin lines – denoted DP).

*with minimum relative error, as well as the iterate that corresponds to the stopping rule as defined above. Figure 4.9 shows the correspondence between the noise level and the number of iterations to the optimal solution, as well as the number of iterations determined by the stopping rule.*

*First, we note that the number of iterations that can reliably be performed decreases when the noise level increases. Furthermore, we note that the behavior of the thin lines corresponding to the stopping rule are similar to the lines for the optimal solutions. This indicates that there indeed for all three methods is a correspondence between the residual norm and the optimal relative error. Due to the value $\nu = 1.1$, the stopping rule should be expected to choose slightly over-regularized solutions which is indeed seen to be the general case (the thin lines generally lies below the thick lines).*

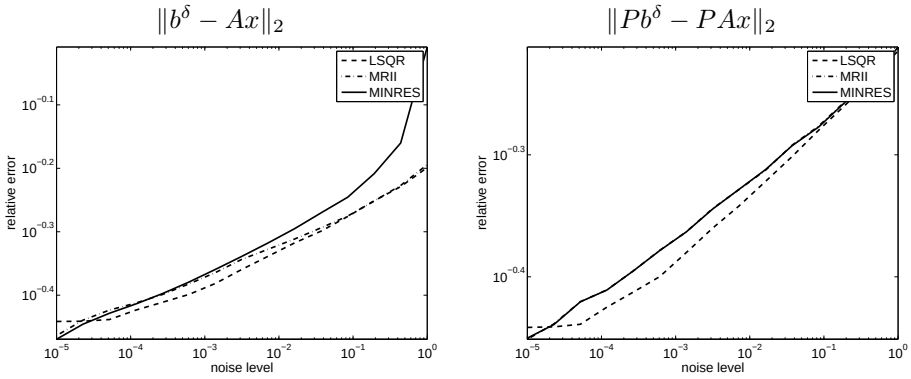



Figure 4.10: Correspondence between amount of noise in the right-hand side and the optimal relative error produced by LSQR, MINRES, and MR-II.

*However, for the nonpermuted problems (the left column of Fig. 4.9) and the higher noise levels (*$\|e\|_2/\|b\|_2 \gtrapprox 10^{-2}$ *and* $\|e\|_2/\|b'\|_2 \gtrapprox 10^{-2}$*) we see that the optimal number of MINRES iterates is* lower *than the number of iterates determined by the discrepancy principle. Obviously, too much noise creeps into the solutions before the residual is reduced enough. Clearly, this is due to the included noise components in the MINRES solution subspace.*

*We note also that MINRES and MR-II applied to* (4.43) *and* (4.44) *behave similarly despite the fact that the convergence for the noise-free right-hand side in Example 4.5 was seen to be faster for MINRES and MR-II. I.e., the noise in the right-hand side severely affects the convergence, but does not spoil the regularizing effect and the use of the stopping rule.*

*Finally, Fig. 4.10 shows the correspondence between the noise level and the optimal relative error for the problems in* (4.43)*. The behavior is very similar for the problems* (4.44)*. We note that there is a clear correspondence between the noise level and the best obtainable solution quality.*

#### 4.3.2.2 Summary

The above series of examples shows how the convergence of MINRES and MR-II depend on the distribution of the eigenvalues of the coefficient matrix, but nevertheless can be considered as regularization methods when applied to a discrete ill-posed problem. We saw also that a stopping rule based on the discrepancy principle seems to make sense for both methods – in agreement with earlier studies [31, 56, 58]. But we also saw that – for higher noise levels – the MINRES iterates may be seriously affected by noise such that the stopping rule determines under-regularized solutions.

### 4.3.3 Nonsymmetric $A$ - GMRES and RRGMRES

Regarding the regularization properties of GMRES and RRGMRES, only a few references exist in the literature, e.g., Calvetti et al. [11] that try to justify theoretically for the regularization properties of GMRES, and Brown and Walker [5] concerning GMRES applied to singular and nearly singular systems. Several other examples where GMRES and RRGMRES are used as regularization methods also exist [6, 10, 13, 14].

Unfortunately, the very limited number of papers in this field reveals a deep lack of understanding of the methods. It is not the ambition with this section to tell the complete tale of GMRES and RRGMRES, but merely to shed some more light on the regularization properties.

The convergence analysis is a bit more well-described. See e.g., [61, 63, 83] for results regarding the convergence of the residual of Krylov subspace methods, including CG, MINRES and GMRES. Unfortunately, we are not particularly interested in the ultimate reduction of the residual, but in early termination of the algorithms to get a hopefully regularized solution. Therefore, not only a worst-case convergence analysis is relevant, but also the actual transient behavior of the solution iterates is of great importance.

In the following, we will look mainly at GMRES, and later provide also examples and conclusions for RRGMRES. Because GMRES does not solve the least squares problem in general, it is natural to follow the lines of [11], and we start by looking at the continuous undisturbed system

$$\mathcal{K}f = g \tag{4.45}$$

where $\mathcal{K} : \mathcal{X} \to \mathcal{X}$ is a compact bounded linear operator as described in Section 2.1. We assume that for every $g \in \mathcal{R}(\mathcal{K})$, there exists a unique solution $f \in \mathcal{X}$. I.e., the operator $\mathcal{K}$ is one-to-one and has an inverse on $\mathcal{R}(\mathcal{K})$. This implies that $\mathcal{N}(\mathcal{K}) = \{0\}$. Furthermore, we look at problems where the inverse of $\mathcal{K}$ is unbounded on $\mathcal{X}$ such that tiny fluctuations of the right-hand side can lead to large variations in the corresponding solution.

According to the statements of Hadamard in Definition 2.1, the above description implies that the first and second criterion are fulfilled, i.e., a solution exists and this solution is unique – at least for the exact data $g$. The third condition is violated because $\mathcal{K}^{-1}$ is unbounded. Now, the right-hand side of the problems we actually want to solve often come from measurements and they are contaminated by noise. We therefore deal with the disturbed system

$$\mathcal{K}f = g^{\delta}. \tag{4.46}$$

This problem might very well violate the second statement of Hadamard if $g^{\delta} \notin \mathcal{R}(\mathcal{K})$, and we do *not* want to solve the system (4.46) exactly, which will most probably lead to a useless solution due to the unbounded inverse. Even if $g^{\delta} \in \mathcal{R}(K)$, the solution may indeed be useless. Instead, we impose regularization and solve another problem where the inverse operator has been substituted by the regularized operator $R_{\alpha}^{\delta}$ as described in Section 2.1. The question is whether GMRES can always provide such

a regularized operator, and the obvious counter example is the down-shift operator for which GMRES will never be able to determine any solution at all.

**Example 4.7** *Let $\mathcal{K} : \mathcal{X} \to \mathcal{X}$ be a compact and bounded linear operator and $\mathcal{X}$ be a Hilbert space. Furthermore, let $\mathcal{K}$ describe the downshift operator such that*

$$\mathcal{K}\left(f_1,\, f_2,\, \ldots\right)^T = \left(0,\, f_1,\, f_2,\, \ldots\right)^T .$$

*Clearly, $\mathcal{N}(\mathcal{K}) = \{0\}$, but for a true solution $f = \left(1,\, 0,\, \ldots\right)^T$ and corresponding right-hand side $g = \left(0,\, 1,\, 0,\, \ldots\right)^T$, GMRES will never breakdown, and never reduce the residual.*

This example indicates the beginning of the trouble. We continue for a little while in the lines of [11] and assume that GMRES is able to solve the noise-free problem (4.45) exactly in $k < \infty$ iterations to avoid obvious problems like the infinite downshift operator. Then we can formulate bounds on the difference between the GMRES iterates of the noise-free problem $f^{(k)}$ and the iterates $f^{\delta(k)}$ using the noisy right-hand side $g^\delta$. Specifically, the difference $\|f^{(k)} - f^{\delta(k)}\|_2 \leq C\delta$, where $\delta$ is the noise level in the measured right-hand side and $C$ is the constant

$$C = 2(2\beta_k + \mu_{k+1})\|\overline{H}_k^\dagger\|_2^2(1 + \tilde{\delta}) + (\beta_k + \beta_{k+1} + 1)\|\overline{H}_k^\dagger\|_2, \tag{4.47}$$

where $\beta_k$, $\beta_{k+1}$, and $\mu_{k+1}$ are constants defined in [11], and $\overline{H}_k$ is the Hessenberg matrix from the partial Arnoldi decomposition (4.13). It is shown [11, Theorem 3.11] that GMRES is a regularization method in the sense that the solution iterate $f^{\delta(m)}$ where $m$ is the iteration number corresponding to a stopping rule similar to the discrepancy principle, fulfills

$$\lim_{\delta \searrow 0} \sup_{\|g - g^\delta\| \leq \delta} \|f - f^{\delta(m)}\| = 0 \ . \tag{4.48}$$

That is, as the noise level decreases, $f^{\delta(k)}$ approximates the true unique solution to (4.45). This definition of a regularization method is very similar to the one in Definition 2.3, that was based on the minimum-norm least squares solution.

Unfortunately, it is problematic to define GMRES as a general regularization method in the above sense. Firstly, the assumption that a unique solution exists is violated if $\mathcal{N}(\mathcal{K}) \neq \{0\}$. And even if $\mathcal{N}(\mathcal{K}) = \{0\}$, the need for discretization will give similar problems because the discrete operator $A$ will have a numerical nullspace. Therefore, the numerical solution of the problem will be extremely unstable even though the continuous formulation in theory has a well-defined solution. This is to some extent mimicked in the constant $C$ in (4.47) by $\|H_{k+1}^\dagger\|_2$ which for increasing $k$ will approximate the norm of $\mathcal{K}^{-1}$ and therefore can be extremely large. In the discrete setting, this potentially leads to numerical problems with extreme ill-conditioning before a decent regularized solution is computed, even though the true continuous problem has no nullspace and in theory has a unique solution.

The assumption that GMRES is able to compute the exact solution of the noise-free problem in a finite number of iterations is also a limitation and can mean one of two things. Either, we are lucky that the exact solution $f \in \mathcal{X}$ actually lies in the subspace $\mathcal{K}_k(\mathcal{K}, g)$; but in general, when the true solution is unknown, we can not rely on this. Or the Hilbert space $\mathcal{X}$ must itself be finite. But if $\mathcal{X}$ is finite then $\mathcal{K}^{-1}$ can only be unbounded if $\mathcal{N}(\mathcal{K}) \neq \{0\}$ — else there would exist a smallest singular value $\sigma_n > \tau$ where $\tau > 0$ is a finite constant, and the inverse will not be unbounded, but have $\sigma_1 \leq 1/\tau$.

Thus to define GMRES as a regularization method based on (4.48) is limiting and applies only to very specific problems. The only viable way in a more general case is to look for some generalized solution to the discrete problem with disturbed right-hand side, and following again the basic regularization theory [22, 29], we seek the least squares solution of minimal norm. But can we expect GMRES to provide a solution like that? First, we restate a theorem due to Brown and Walker [5].

**Theorem 4.9** *GMRES determines a least squares solution of* (2.4) *without breakdown for all* $b$ *and* $x^{(0)}$ *if and only if* $\mathcal{N}(A) = \mathcal{N}(A^T)$. *If* $\mathcal{N}(A) = \mathcal{N}(A^T)$ *and a least squares solution is reached at some step, then GMRES breaks down at the next step, with breakdown through degeneracy of the Krylov subspace if* (2.4) *is consistent and through rank deficiency of the least squares problem* (4.15) *otherwise. Furthermore, if* (2.4) *is consistent and* $x^{(0)} \in \mathcal{R}(A)$, *then the solution reached is the minimum norm solution.*

Proof. See Brown and Walker [5]. □

Theorem 4.9 shows that $A$ must fulfill certain properties for GMRES to give even a least squares solution and this limits the use of the general theoretical definition of a regularization method where the regularized inverse should give the minimum-norm least squares solution when the noise level goes to zero. As usual, we assume that the initial iterate $x^{(0)} = 0$, and Theorem 4.9 shows that the requirements for GMRES to compute a least squares solution of minimum norm is that $\mathcal{N}(A) = \mathcal{N}(A^T)$ which implies that $\mathcal{R}(A) = \mathcal{R}(A^T)$, i.e., that $A$ is *range symmetric*. Moreover, the problem must be consistent with $b^\delta \in \mathcal{R}(A)$.

If we turn to the RRGMRES method, this was developed to solve inconsistent systems [7]. Indeed it deals with the last requirement and computes the minimum-norm least squares solution also for inconsistent problems as long as $\mathcal{R}(A) = \mathcal{R}(A^T)$.

Now compare with Table 4.2. Indeed for range symmetric problem we note that GMRES (when $b^\delta \in \mathcal{R}(A)$) and RRGMRES in general fall in the top category and produce solutions in a subspace of $\mathcal{R}(A^T)$ similar to CGLS and LSQR. But we also know from the SVD analyses that the solution iterates for both GMRES (4.30) and RRGMRES (4.31) are based on the nondiagonal mixing matrix $C$ from (4.28).

In the following we study two different types of problems. Both types have small singular values that cluster at machine precision, but their behavior varies. A *numerically rank-deficient problem* has a singular value spectrum with a clear gap between

the "large" singular values and the "small". Therefore, it is natural to define a "good" subspace corresponding to the numerical range of $A$, $\mathcal{R}(A)$, a "bad" subspace corresponding to the numerical nullspace of $A$, $\mathcal{N}(A)$, as well as the numerical rank $r$ of $A$. Indeed, if the singular components corresponding to the "good" subspace are not too affected by noise, then the standard minimum-norm least squares solution $x_{\text{ls}} = A^{\dagger}b^{\delta}$ will be a decent solution. This solution is actually the TSVD solution $x_k$ corresponding to the regularization parameter $k = r$.

A *discrete ill-posed problem*, on the other hand, has singular values that decay gradually to machine precision $\epsilon$ without any gap in the spectrum. We can again define the numerical rank $r$ of $A$ such that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq \epsilon$ and $\sigma_{r+1} \approx \ldots \sigma_n \approx \epsilon$, and we define the corresponding numerical subspaces $\mathcal{R}(A^T) = \text{span}\{v_1, \ldots, v_r\}$ and $\mathcal{R}(A) = \text{span}\{u_1, \ldots, u_r\}$. But due to the noisy right-hand side $b^{\delta}$, we cannot assume that all right-hand side components in $\mathcal{R}(A)$ are not too affected by noise. In this case the TSVD solution $x_r$ will probably be severely contaminated by inverted noise. It also means that we cannot put a solution in the entire $\mathcal{R}(A^T)$. We note that a rank-deficient problem for which the noise level severely affects the right-hand side components in $\mathcal{R}(A)$ will effectively behave like a discrete ill-posed problem.

#### 4.3.3.1 Discrete Rank-Deficient Problem

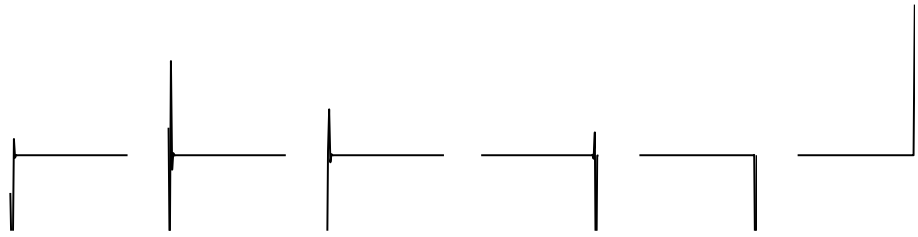
First, we look at the case where $A$ is numerically rank-deficient. In this case Theorem 4.9 applies, and if $\mathcal{N}(A) = \mathcal{N}(A^T)$ then RRGMRES will find a least squares solution of minimum norm, and so will GMRES if $b^{\delta} \in \mathcal{R}(A)$. The examples provided in [5] illustrate well the performance of GMRES for singular systems, but here we consider a standard test problem from the inverse problems community.

**Example 4.8** *We consider the inverse heat equation [15], and use the specific implementation* heat *from MOORe Tools [49]. The coefficient matrix $A \in \mathbb{R}^{100\times 100}$ is a discretization of a first kind Volterra integral equation with the kernel*

$$K(s,t) = \frac{1}{2\sqrt{\pi(s-t)^3}} e^{-\frac{1}{4t}}.$$

*The matrix $A$ is numerically rank-deficient with $\sigma_1 > \cdots > \sigma_{97} > 10^{-7}$ and $\sigma_{98} \approx \sigma_{99} \approx \sigma_{100} \approx 10^{-15}$. We consider the noise-free system $Ax = b$ where $b$ is the noise-free right-hand side, $b = Ax_{\text{exact}} \in \mathcal{R}(A)$ and note that $\mathcal{N}(A) = \text{span}\{v_{98}, v_{99}, v_{100}\} \neq \mathcal{N}(A^T) = \text{span}\{u_{98}, u_{99}, u_{100}\}$. Figure 4.11 illustrates the vectors that span the two nullspaces; obviously the left nullspace vectors live in the left side of the domain, and the right nullspace vectors live in the right side.*

*We perform 40 GMRES and RRGMRES iterations, as well as 40 CGLS iterations without reorthogonalization. Figure 4.12 (top) shows the explicitly computed residuals of the iterates, and we note that neither GMRES nor RRGMRES are able to reduce the residual as much as CGLS. The bottom plot in Fig. 4.12 shows the relative errors, and it is interesting to note that neither GMRES nor RRGMRES are able to reduce the relative error at all – even though the residuals are of course nonincreasing. On*



Left nullspace vectors $u_{98}$, $u_{99}$, $u_{100}$ Right nullspace vectors $v_{98}$, $v_{99}$, $v_{100}$

Figure 4.11: Illustration of the vectors spanning the left and right nullspaces of the coefficient matrix $A$ for the inverse heat problem.

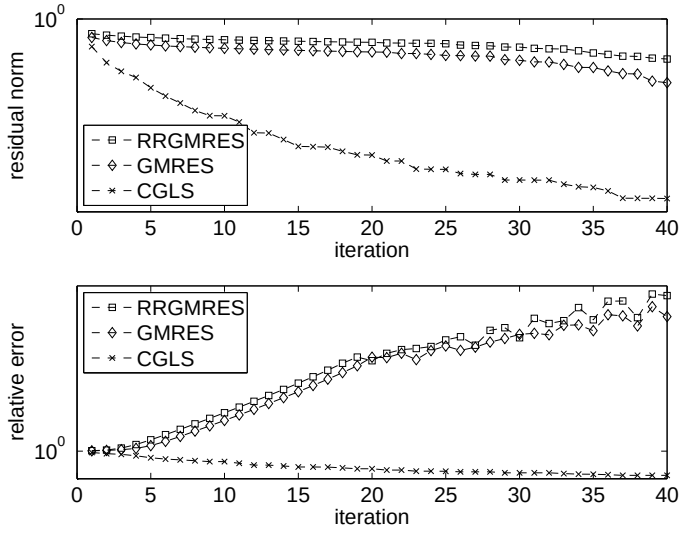



Figure 4.12: Top: reduction of the residual norm for 40 GMRES, RRGMRES and CGLS iterates applied to the inverse heat test problem. Bottom: relative errors of the solution iterates compared to the true solution $x_{\text{exact}}$ of the inverse heat problem.

*the other hand, CGLS slowly reduces the relative error. The reason for this unwanted behavior is obvious. The residual is larger for GMRES and RRGMRES because components needed to construct the least squares solution lie in* $\mathcal{N}(A^T)$ *and are never included in the solution subspaces, i.e.,* $\mathcal{R}(A^T) \cap \mathcal{N}(A^T) \neq \{0\}$*. Similarly, unwanted components from* $\mathcal{R}(A) \cap \mathcal{N}(A) = \{0\}$ *that potentially increase the solution norm and the relative error dramatically can be included in the solution iterates without increasing the residual. Figure 4.13 shows the first five GMRES iterates, and we note*

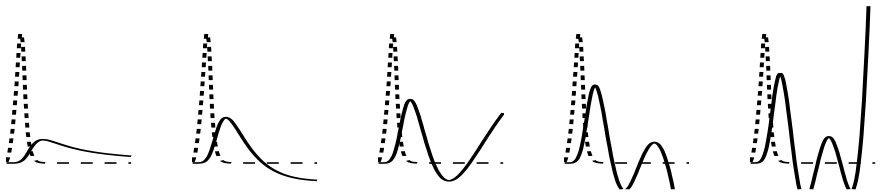


Figure 4.13: The first five GMRES solution iterates (solid lines) and the true solution $x_{\text{exact}}$ (dashed lines) for the inverse heat test problem.

*that they tend to explode near the right boundary due to inclusion of components from the right nullspace. We note that the RRGMRES iterates (not shown here) behave similarly.*

Based on Theorem 4.9 it is clear why the inverse heat problem is troublesome. But the situation gets even more complicated when dealing with discrete ill-posed problems, and especially noisy ill-posed problems.

#### 4.3.3.2 Discrete Ill-Posed Problems

Assume that a matrix $A \in \mathbb{R}^{n\times n}$ has the SVD $A = U\Sigma V^T$ and that the singular values are steadily decreasing, but strictly positive, i.e., $\sigma_1 > \sigma_2 > \cdots > \sigma_n > 0$. Furthermore, let $C = V^T U$ as in (4.28). Finally, define the coefficients $\xi_i = v_i^T x$ for $i = 1, \ldots, n$, and write the noise free right-hand side (the first Krylov vector for GMRES) as

$$b = Ax = U\Sigma V^T x = \sum_{i=1}^{n} \sigma_i \xi_i u_i = \sum_{i=1}^{n} \sigma_i \xi_i V c_i, \tag{4.49}$$

where the left singular vectors are expressed in terms of the right singular vectors $u_i = Vc_i$ and $c_i$ is the $i$th column of $C$. Obviously, the coefficient $\xi_i$ that express the $v_i$ component of the true solution is scaled and expressed as a combination of right singular vectors, corresponding to $u_i$. We now look at the second GMRES Krylov vector

$$Ab = U\Sigma V^T b = \sum_{k=1}^{n} \sigma_k \left( \sum_{i=1}^{n} \sigma_i \xi_i c_{k,i} \right) V c_k, \tag{4.50}$$

and note two problems. As for the right-hand side vector $b$, the $k$th component $u_k = Vc_k$ is a combination of right singular vectors. But in addition, the $k$th scaling factor $\sigma_k \sum_{i=1}^{n} \sigma_i \xi_i c_{k,i}$ depends on the $k$th singular value multiplied by a weighted sum of possibly all the singular values. That is, the generated Krylov vector may not primarily consist of the left singular vectors corresponding to the largest singular

values. Now consider a noisy right-hand side $b^\delta = b + e$ where $e$ is white noise

$$b^\delta = Ax + e = \sum_{i=1}^{n} \sigma_i \xi_i u_i + UU^T e = \sum_{i=1}^{n} (\sigma_i \xi_i + u_i^T e) u_i, \tag{4.51}$$

and recognize $\sigma_i \xi_i$ as $u_i^T b$. As noted in Section 2.2.1, the discrete Picard condition implies that the coefficients corresponding to low indices $i$ will be dominated by true solution components, whereas the latter will be dominated by the noise. We want to reconstruct the solution in a subspace in $\mathcal{R}(A^T) = \text{span}\{v_1, \ldots, v_r\}$ corresponding to the components above the noise level. That is, we want to compute a regularized solution $x_\alpha$ such that $Ax_\alpha$ approximates the part of $b^\delta$ that is not too affected by noise. This is reflected in the discrepancy principle, where we want to reconstruct the solution such that the residual $\|b^\delta - Ax_\alpha\|_2$ is at a level corresponding to the noise level. But obviously, when we construct the Krylov vector $Ab^\delta$, see (4.50), we mix not only the singular components in the solution subspace, but also the noise components in the scaling factors.

Consider now a severely nonsymmetric problem. Let $A \in \mathbb{R}^{n \times n}$ have singular values that decay equidistantly in log scale, and let the smallest singular value $\sigma_n > \epsilon$ be larger than machine precision. Moreover, let the relationship between the singular basis vectors $U = \big(u_1, \ldots, u_n\big)$ and $V = \big(v_1, \ldots, v_n\big)$ be given as

$$u_i = v_{n-i+1}, \quad \text{for} \quad i = 1, \ldots, n.$$

Assume that we have a true solution $x_{\text{exact}} \in \mathbb{R}^n$ and the true undisturbed right-hand side $b = Ax_{\text{exact}}$. We follow Eq. (4.49) and write the right-hand side as

$$b = \sum_{i=1}^{n} \sigma_i \xi_i u_i = \sum_{i=1}^{n} \sigma_i \xi_i v_{n-i+1}.$$

Due to the inverse order of the left and right singular vectors $u_i$ and $v_i$, as well as the decreasing singular values $\sigma_i$, we see that $b$ consists mainly of the latter right singular vectors. Following (4.50) we write also the second GMRES Krylov vector $Ab$ (i.e., the first RRGMRES vector) as

$$Ab = \sum_{k=1}^{n} \sigma_k v_k^T \left( \sum_{i=1}^{n} \sigma_i \xi_i v_{n-i+1} \right) v_{n-k+1}^T = \sum_{k=1}^{n} \sigma_k \sigma_{n-k+1} \xi_{n-k+1} v_{n-k+1}, \tag{4.52}$$

where the last equality holds due to orthogonality of the $v_i$s. We observe that small singular values will affect all components of the sum. The weights are – in addition to the solution components $\xi_i$ – controlled by the products $\sigma_k \sigma_{n-k+1}$. For the steadily decreasing singular values, this filter will possibly include all right singular components $v_1, \ldots, v_n$ in the Krylov subspace with similar weight. Therefore, GMRES may in two iterations (and RRGMRES in one) be able to construct a solution $\hat{x}^{(2)}$ (and $\bar{x}^{(1)}$) in a two-dimensional subspace of $\text{span}\{v_1 \ldots, v_n\}$. In case of a noisy right-hand side (4.51), this solution might also have components that are dominated by noise. The following example shows how this behavior can be observed for GMRES.

**Example 4.9** *We construct a problem based on the coefficient matrix* $A \in \mathbb{R}^{100\times 100}$ *and the true discrete solution* $x_{\text{exact}} \in \mathbb{R}^n$ *from the* deriv2 *test problem from MOORe Tools [49]. Let* $A = U\Sigma V^T$ *be the SVD of* $A$ *and define* $\widehat{U} = (\,\hat{u}_1, \ldots, \hat{u}_n\,)$ *where* $\hat{u}_i = u_{n-i+1}$ *for* $i = 1, \ldots, n$. *Furthermore, let the singular values* $\hat{\sigma}_i = 10^{-4(i-1)/(n-1)}$ *such that they decay equidistantly in log-scale from* $10^0$ *to* $10^{-4}$, *and define the diagonal matrix* $\widehat{\Sigma} = \text{diag}(\,\hat{\sigma}_1, \ldots, \hat{\sigma}_n\,)$. *A new non-symmetric and fairly well-conditioned coefficient matrix is now given by* $\widehat{A} = \widehat{U}\widehat{\Sigma}V^T$, *and the undisturbed right-hand side is computed as* $b = \widehat{A}x_{\text{exact}}$. *Finally, we define also the noisy right-hand side* $b^\delta = b + e$ *where* $e$ *is white Gaussian noise scaled such that the noise level is* $\|e\|_2/\|b\|_2 = 10^{-2}$.

*The solution* $x = \widehat{A}^{-1}b$ *will be a good approximation to* $x_{\text{exact}}$ *because* $A$ *is fairly well-conditioned. But due to the noise in* $b^\delta$, $x^\delta = \widehat{A}^{-1}b^\delta$ *is not a good approximation to* $x_{\text{exact}}$. *Therefore we need to regularize the noisy problem.*

*We compute 100 iterates of GMRES and LSQR applied to both the noise-free and the noisy problem. Figure 4.14 shows three plots as described in the following*

- *The relative error of the iterates when applied to the noise free problem, compared to the true solution* $x_{\text{exact}}$.
- *The relative error of the iterates when applied to the noisy problem, compared to the true solution* $x_{\text{exact}}$.
- *The relative error of the iterates when applied to the noisy problem, compared to the noisy solution* $x^\delta$.

*It is clear that GMRES converges fast to the solutions* $x_{\text{exact}}$ *and* $x^\delta$ *for the noise-free and noisy problem, respectively. Therefore, GMRES does* not *exhibit semi-convergence when applied to the noisy problem, as seen from the middle plot in Fig. 4.14. On the other hand, LSQR exhibits semi-convergence and in the noisy case approximates first the true exact solution* $x_{\text{exact}}$ *before heading off for the noisy solution* $x^\delta$.

The above example shows that GMRES in few iterations is able to compute the solution to both the noise-free and the noisy problem. That is, GMRES does not necessarily first include components that are not too affected by noise before it includes the more noisy components. Due to the mixing in the second Krylov vector $Ab$ (4.52) of the GMRES subspace (and due to the specific right-hand side components for this problem) all singular components are included in the Krylov subspace. It is therefore possible to approximate the wanted solution for the noise-free problem, but also the noisy solution for the noisy problem. We stress that the above example serves only as an illustration of GMRES, and due to the reverse order of the left singular vectors, it does not particularly resemble an inverse problem. Therefore, we now turn to look at some test problems of more inverse nature.

GMRES and RRGMRES have been used to provide regularized solutions to several types of inverse problems, see, e.g., [6, 8, 10, 11]. In the following, we will first look at a few examples where both GMRES and RRGMRES seem to work.

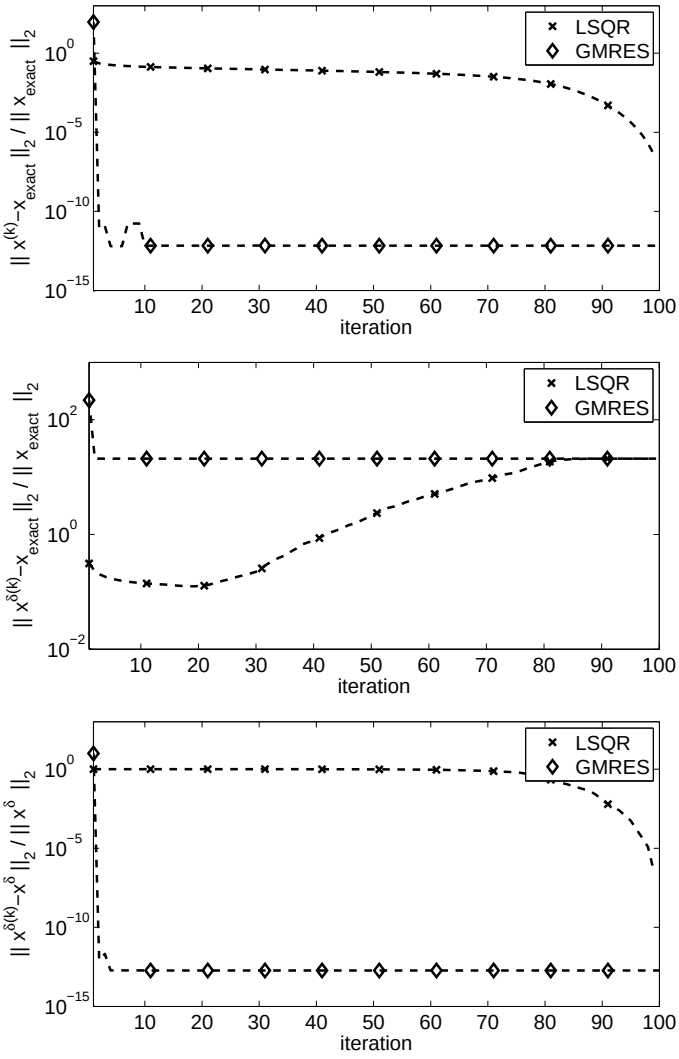
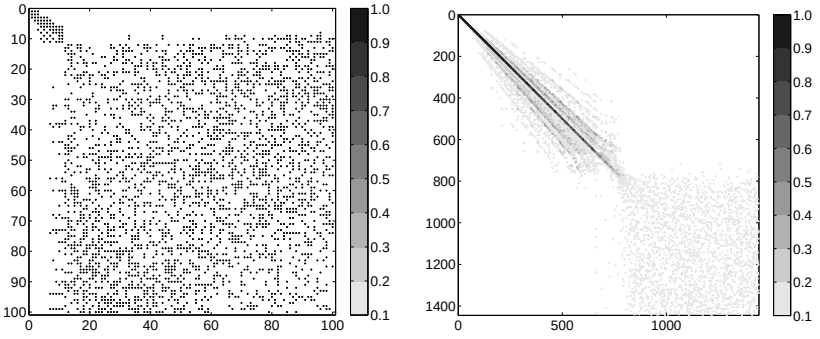



Figure 4.14: Relative errors for GMRES and LSQR for flipped deriv2 problem.

**Example 4.10** *We use the non-symmetric test problem* baart *from MOORe Tools [49] with* $A \in \mathbb{R}^{100\times 100}$*, the given true solution* $x_{\text{exact}}$ *and the noise-free right-hand side* $b$*, and let* $A = U\Sigma V^T$ *be the SVD. Figure 4.15 (left) shows the overall structure of the matrix* $C$ *from* (4.28)*. For clarity, the elements have been binned and gray scale color coded according to the colorbar. It is seen that the upper left part of the plot (corresponding to the singular values above machine precision) is indeed not*




Example 4.10 (baart) Example 4.11 (image)

Figure 4.15: Illustration of the mixing matrix $C = V^T U$ for the baart test problem in Example 4.10 and the image test problem in Example 4.11.

*diagonal, i.e., the left and right singular bases are indeed mixed. Noise is added such that* $\|e\|_2/\|b\|_2 = 10^{-4}$*, and Fig. 4.16 shows the residual norms and relative errors for the first 8 GMRES, RRGMRES, and LSQR iterates. The residual plot shows also a dashed line that indicate a level connected to the discrepancy principle (the stopping rule from Definition 4.8 with* $\nu = 1.1$*). We see that both GMRES and RRGMRES construct solutions with lower relative errors than the LSQR solutions, and furthermore that the best solutions correspond to iterates that are well determined by the stopping rule.*

*The GMRES and RRGMRES subspaces definitely seem to be better than the LSQR subspace, and we note from Table 4.2 that while LSQR constructs solutions in* $\mathcal{R}(A^T)$*, then both GMRES and RRGMRES construct solutions in* $\mathcal{R}(A)$*. We therefore measure how well the sought solution* $x_{\text{exact}}$ *fits in* $\text{span}\{u_1, \ldots, u_p\}$ *and* $\text{span}\{v_1, \ldots, v_p\}$ *for* $p = 1, \ldots, 20$*, i.e., how well the solution is represented in subspaces of increasing dimension, spanned by the first left and right singular vectors, respectively. We let*

$$\begin{aligned} \theta_p^{(v)} &= \angle\left(x_{\text{exact}}, \text{span}\{v_1, \ldots, v_p\}\right) \\ \theta_p^{(u)} &= \angle\left(x_{\text{exact}}, \text{span}\{u_1, \ldots, u_p\}\right) \end{aligned} \qquad p = 1, \ldots, 20, \tag{4.53}$$

*denote the smallest angles between* $x_{\text{exact}}$ *and the subspaces of increasing dimension, and show* $\theta_p^{(v)}$ *and* $\theta_p^{(u)}$ *in Fig. 4.17. This plot reveals that the left singular vectors provide a basis that is much better suited for representing this particular solution.*

The GMRES and RRGMRES methods have also been used in different settings for the solution of image deblurring problems, see, e.g., [6, 10]. We consider now a test problem with a nonsymmetric PSF matrix for which GMRES has been shown to provide nice regularized solutions.

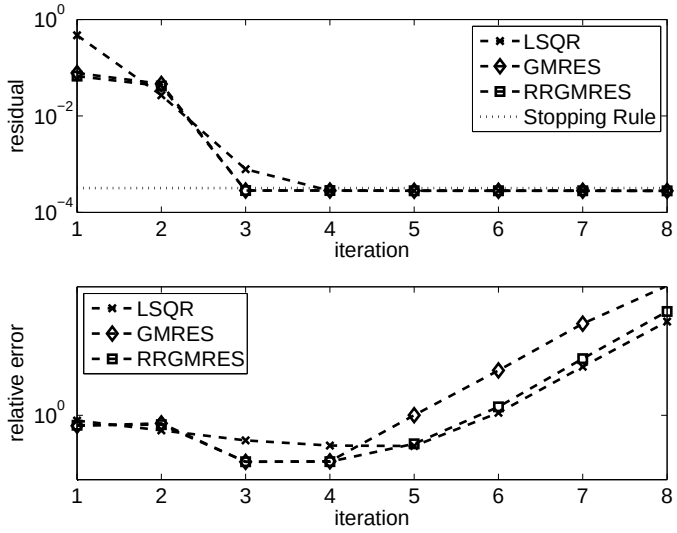



Figure 4.16: Residuals norms and relative errors for GMRES, RRGMRES and LSQR iterates for the baart test problem.

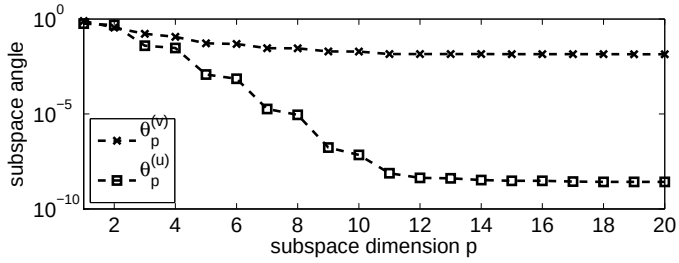



Figure 4.17: Angles between the true wanted solution and the span of an increasing number of left and right singular vectors.

**Example 4.11** *We consider a setting similar to the one used in [10, Example 4.3]. First, we model a spatially variant PSF by combining two PSF matrices that describe two different spatially invariant PSFs. Construct* $A_1$, $A_2 \in \mathbb{R}^{n^2 \times n^2}$ *as Kronecker products such that* $A_p = T_p \otimes T_p$ *implement Gaussian blur as described in* (3.9). *Now let the PSF matrix* $A \in \mathbb{R}^{n^2 \times n^2}$ *be constructed as* $A = I_1 A_1 + I_2 A_2$ *where* $I_1$, $I_2 \in \mathbb{R}^{n^2 \times n^2}$ *are given as*

$$I_1 = \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} \quad , \quad I_2 = \begin{pmatrix} 0 & 0 \\ 0 & I \end{pmatrix},$$

*and* $I$ *is the identity matrix of size* $(n^2/2) \times (n^2/2)$. *The PSF matrix* $A$ *so defined is nonsymmetric, and the example from [10] shows that for a problem with* $n = 512$, *and a natural image, GMRES is able to produce regularized solutions (superior to the CGLS solutions) to a linear system* $Ax = b$.

*Now we let* $n = 38$, *and* $\sigma_s = 4$ *and* $\sigma_t = 4.5$. *For a problem of this size, we can explicitly calculate the SVD of* $A$ *and construct the matrix* $C = V^T U$ *from* (4.28). *The right plot in Fig. 4.15 shows the structure of* $C$. *We note that even though* $A$ *is nonsymmetric, the matrix* $C$ *that mixes the singular components in the GMRES solution subspace is close to diagonal in the upper left corner which corresponds to the most significant singular components. Therefore, GMRES and RRGMRES will tend to initially generate a favorable solution subspace and these methods are therefore expected to (at least initially) compute regularized solutions as concluded in [10, Example 4.3].*

The two examples above show two different cases for which GMRES and RRGMRES are indeed able to produce regularized solutions to a discrete ill-posed problem. Example 4.10 illustrates a case where the wanted solution is (by chance) better expressed in a subspace of the left singular vectors than the right. This gives the subspaces of GMRES and RRGMRES an advantage compared to the LSQR and CGLS subspace – and also compared to the subspaces underlying TSVD and Tikhonov regularization. On the other hand, one cannot conclude in general that GMRES and RRGMRES always behave well if the sought solution is better represented in the left singular basis because also the noise and the unstructured damping of the subspace components influence the generated subspaces. Example 4.11 shows another case where the mixing of the singular components is so small that $C$ is close to diagonal. This implies that the first left and right singular vectors corresponding to the largest singular values, are similar. Therefore, GMRES and RRGMRES tend to produce solution subspaces with a regularizing effect like LSQR and CGLS, similar to the case for MINRES and MR-II for symmetric problems.

Lemma 4.6 showed that for real normal matrices, the mixing of the singular components expressed by $C$ from 4.28 is limited to a mixing within subspaces of dimension two. As seen above, a matrix $C$ that is not too far from diagonal may indicate that GMRES and RRGMRES are able to produce regularized solutions. We now look at an ill-posed problem with a normal coefficient matrix.

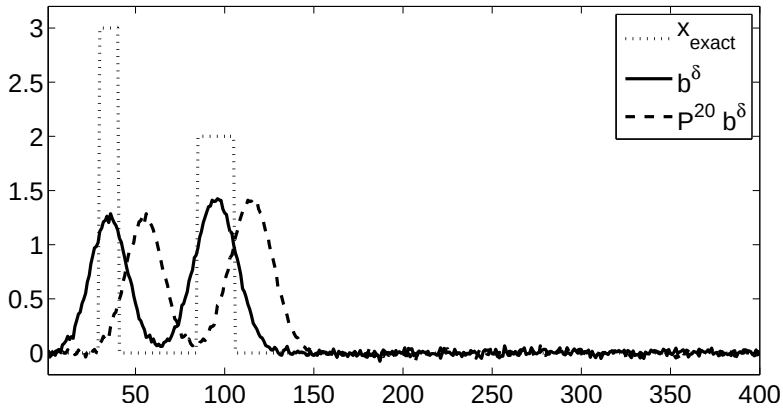



Figure 4.18: True solution $x_{\text{exact}}$ to normal matrix example, and the two right-hand sides $b^\delta$ and $P^{20}b^\delta$.

**Example 4.12** *Let the symmetric $A \in \mathbb{R}^{400}$ be an implementation of a one-dimensional Gaussian blur, similar to the two-dimensional blur from the MINRES examples. Furthermore, consider periodic boundary conditions such that $A$ is circulant. Define now the circulant downshift operator $P$ and let a second circulant coefficient matrix be defined by $P^{20}A$, i.e., by applying 20 downshifts to $A$. Both $A$ and $P^{20}A$ are normal matrices with the same singular values. We construct a true solution $x_{\text{exact}}$ as seen in Fig. 4.18 which also shows the two noisy right-hand sides $b^\delta$ and $P^{20}b^\delta$ where the additive white noise $e$ is scaled such that $\|e\|_2/\|Ax_{\text{exact}}\|_2 = 5 \cdot 10^{-2}$. We use GMRES, RRGMRES and LSQR to solve the two equivalent problems*

$$\min_x \|b^\delta - A\,x\|_2 \quad \text{and} \quad \min_x \|P^{20}b^\delta - P^{20}A\,x\|_2 \ .$$

*Figure 4.19 shows the convergence histories when applying the iterative methods to the two problems. Obviously, the convergence of both GMRES and RRGMRES are severely affected by the downshift operator, whereas LSQR of course is not affected by the orthogonal transformation of the least squares problem. The plot also shows a dashed line that indicate a level connected to the discrepancy principle as defined by the stopping rule in Definition 4.8 with $\nu = 1.1$.*

*Compare now the convergence history of the residual with the relative errors of the solution iterates shown in Fig. 4.20. For the first problem $\min \|b^\delta - Ax\|_2$, all three methods exhibit semi-convergence, and the optimal solutions correspond quite well to iterates for which the residual drop below the level determined by the stopping rule. On the other hand, for the second problem $\min \|P^{20}b^\delta - P^{20}Ax\|_2$ we see that the relative errors for RRGMRES and especially GMRES do not exhibit strict semi-convergence.*

*Consider now Table 4.4. Here we see that for the original problem $\min \|b^\delta - Ax\|_2$*

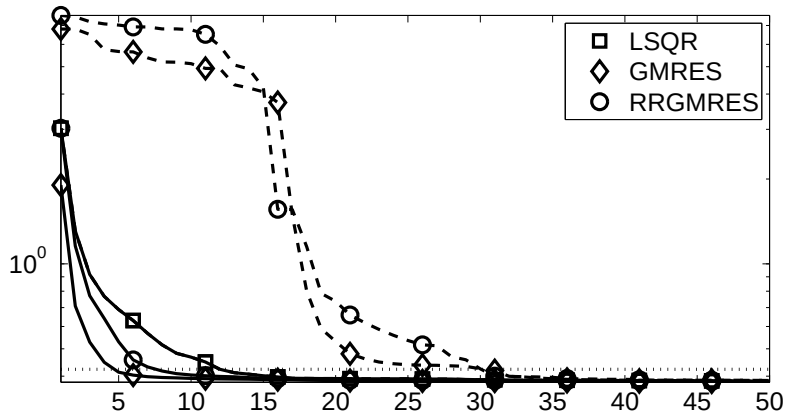



Figure 4.19: Residuals for the iterates of LSQR, GMRES, and RRGMRES when applied to $Ax = b^\delta$ (solid lines), and $P^{20}Ax = P^{20}b^\delta$ (dashed lines). Also shown is a level connected to the stopping rule in Definition 4.8 with $\alpha = 1.1$ (dotted line).

| | LSQR | GMRES | RRGMRES |
|---|---|---|---|
| $\min_x \|b - A\,x\|_2$ | 0.4291 (16) | 0.4967 (3) | 0.4270 (13) |
| $\min_x \|P^{20}b - P^{20}A\,x\|_2$ | 0.4291 (16) | 0.4402 (32) | 0.4427 (31) |

Table 4.4: Relative errors and number of iterations for best solution iterates when applying LSQR, GMRES, and RRGMRES to the linear systems with normal coefficient matrices.

*for which $A$ is actually symmetric, the optimal solutions and the number of iterates follow the similar case from Example 4.3. That is, GMRES produces a worse solution than LSQR, but in very few iterations, and RRGMRES produces a solution similar to the LSQR solution in fewer iterations. On the other hand, we see that for the problem* $\min \|P^{20}b^\delta - P^{20}A\,x\|_2$*, both GMRES and RRGMRES require twice as many iterations as LSQR, and furthermore produce solutions that are slightly worse than the optimal LSQR solution.*

The example above illustrates that for a discrete ill-posed problem with a normal coefficient matrix for which $C = V^TU$ is close to diagonal, GMRES and RRGMRES need not exhibit strict semi-convergence. Therefore, it might be difficult in general to provide optimal stopping rules. On the other hand, the solution subspaces will eventually include components that are needed to reconstruct the regularized solution because trivially all normal matrices are range symmetric, i.e., $\mathcal{R}(A) = \mathcal{R}(A^T)$.

The final example of this section shows a more general nonnormal and nonsymmetric test problem for which neither GMRES nor RRGMRES produce regularized

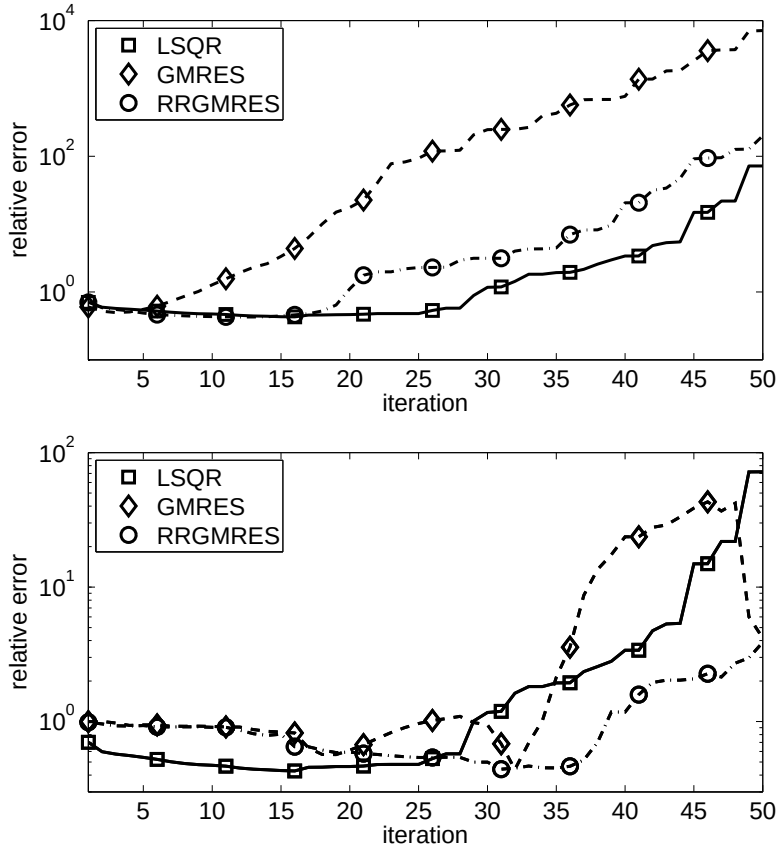



Figure 4.20: Relative errors for iterates of LSQR, GMRES, and RRGMRES applied to $\min \|b^\delta - Ax\|_2$ (top) and $\min \|P^{20}b^\delta - P^{20}Ax\|_2$ (bottom).

solutions. Furthermore, the example shows that providing general stopping rules is difficult, if not meaningless.

**Example 4.13** *We use the nonsymmetric, discrete ill-posed problem* ilaplace *from MOORe Tools [49] and use the first implemented solution. The singular values of* $A \in \mathbb{R}^{100\times100}$ *decay gradually towards zero and hit machine precision around index 30, i.e.,* $\sigma_i \approx 10^{-16}$ *for* $i = \{30, 31, \ldots, 100\}$. *The noise vector* $e$ *contains white Gaussian noise, and 100 different scalings are used such that* $\|e\|_2/\|Ax_{\text{exact}}\|_2 \in \{10^{-10}, 10^{0}\}$. *For each noise level we compute 30 iterates of GMRES, RRGMRES, as well as 30 iterates of LSQR using reorthogonalization of the Krylov vectors. The iterates with optimal relative error are found for all three methods and all noise levels.*

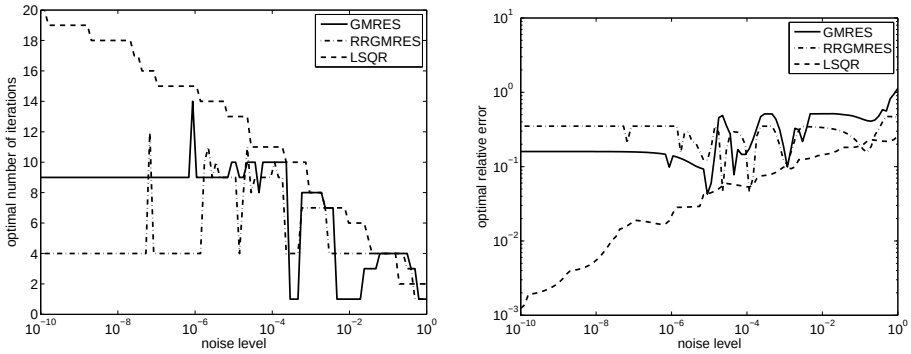



Figure 4.21: Left: index of optimal solution iterate as a function of the noise level when applying LSQR, GMRES, and RRGMRES iterations to the ilaplace test problem. Right: optimal relative error $\|x - x^{(k)}\|_2 / \|x\|_2$ for LSQR, GMRES, and RRGMRES as a function of the noise level.

*Figure 4.21 (left) shows the index of the best solution iterate as a function of the noise level, and Fig. 4.21 (right) shows the corresponding relative errors as a function of the noise level.*

*We observe that for LSQR there is a clear correspondence between the noise level and the optimal solution iterate: for a high noise level only few iterations can be reliably performed before the noise distorts the solutions, whereas more iterations can be performed for a lower noise level. Also, the plot of the relative errors indicate that overall the solutions get better for lower noise levels. But the same is* not *true for GMRES and RRGMRES. For small noise levels, the solutions do not get better than a certain level, and for higher noise levels the number of iterations that can reliably be performed is unpredictable. E.g., for a noise level of* $10^{-10}$*, the 4th RRGMRES iterate is the optimal solution with relative error 0.3500 but for a higher noise level of* $1.15 \cdot 10^{-4}$ *the best RRGMRES iterate is the 10th with a relative error of only 0.04615.*

As mentioned earlier, the sensitivity to the noise is indeed very different for GMRES and RRGMRES than for CGLS and LSQR. Indeed, this behavior and Example 4.13 indicate that it may be very difficult to find suitable stopping rules for GMRES and RRGMRES. Especially that a stopping rule based on the discrepancy principle is of no use.

#### 4.3.3.3 Summary

This section provided insight into some of the properties of GMRES and RRGMRES when applied to rank-deficient and discrete ill-posed problems. The general observation is that both methods *may* produce regularized solutions, but that one should be

very careful indeed to rely on the regularization properties in general. Furthermore, several of the examples illustrate that semi-convergence is not necessarily observed, and that there might be no correspondence between the reduction of the residual and the behavior of the solution iterates. Therefore, the construction of stopping rules is similarly difficult.

### 4.3.4 GMRES and RRGMRES for Nonsquare Systems

In [8], GMRES and RRGMRES were applied to nonsquare systems by augmenting the coefficient matrix and the right-hand side. Assume that we want to use GMRES or RRGMRES to solve an underdetermined system $\min \|b - Ax\|_2$ where $A \in \mathbb{R}^{m\times n}$ with $m < n$. We apply the approach from Section 4.2.3 and obtain the new system

$$\min \|\hat{b} - \widehat{A}x\|_2,$$

where the augmented matrix $\widehat{A} \in \mathbb{R}^{n\times n}$ is square. The last $n - m$ rows are zero, and $\widehat{A}$ is therefore in general not symmetric. Now, let us look at the Krylov subspace used by GMRES and RRGMRES, $\mathcal{K}_k(\widehat{A}, \hat{b})$ and $\mathcal{K}_k(\widehat{A}, \widehat{A}\hat{b})$, respectively. Because of the zero-padding of $\hat{b}$ and $\widehat{A}$ all vectors that span these solution subspaces will obviously be zero for the elements $m$ to $n$. That is, any solution component different from zero in this part of the domain can neither be reconstructed by GMRES nor RRGMRES using this approach.

The example that illustrates the performance of this approach in [8] is the ilaplace test problem from Regularization Tools, and it has exactly the property that the true solution is zero in this part of the domain. The performance of this approach therefore depends crucially on the nature of the true solution – i.e., that the zero-padding corresponds to zeros in the sought solution. Of course, the zero-padding need not be restricted to the lower part of $\widehat{A}$ and $\hat{b}$; the zero rows can be inserted anywhere to make the system square. We illustrate this by an example.

**Example 4.14** *This example deals with interpolation in two dimensions. The operator $A \in \mathbb{R}^{m\times n}$ that maps points on a regular grid of size $n_x \times n_y$ to $m$ measured points on an irregular grid is constructed by the* interpolate *function from MOORe Tools. We create a true target surface inspired by the example in [77, §4], but here the domain is $d_1, d_2 \in [-3\,;\,3]$, i.e., the true surface $u(d_1, d_2)$ is given by*

$$u(d_1, d_2) = \cos(3\pi((d_1/3)^2 + (d_2/3)^2)/4) + \cos((d_1/3)\pi)\sin((d_2/3)\pi).$$

*The true solution $X_{\text{exact}} \in \mathbb{R}^{n_x\times n_y}$ is sampled from $u(d_1, d_2)$ and stacked such that $x_{\text{exact}} = \text{vec}\,(X_{\text{exact}})$ is a vector of length $n_x n_y$. The irregularly placed measurement points are then artificially constructed by $b^\delta = Ax_{\text{exact}} + e$ where the white Gaussian noise $e$ is scaled such that $\|e\|_2/\|Ax_{\text{exact}}\|_2 = 10^{-2}$.*

*We construct an underdetermined system by choosing $m = 200$ and $n_x = n_y = 20$. A three-dimensional illustration of the surface is seen in Fig. 4.22 (left), and the right plot shows a contour plot where the irregularly placed measurements points are shown by small dots. We also construct two additional square coefficient matrices by*

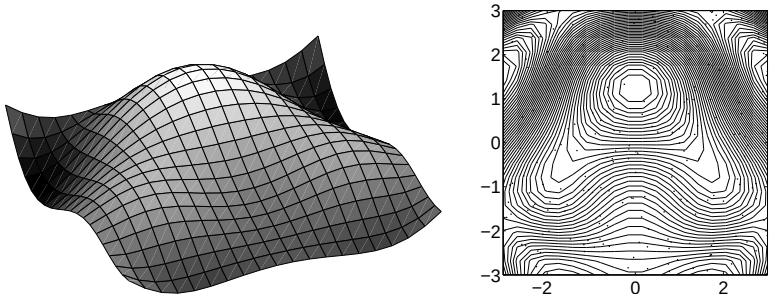



Figure 4.22: Left: three-dimensional illustration of the solution to the inverse interpolation test problem. Right: contour plot of the true solution with dots that indicate the irregularly places measurement points.

$$\widehat{A} = \begin{pmatrix} A \\ Z_{200} \end{pmatrix} \quad \text{and} \quad \widetilde{A} = \begin{pmatrix} Z_{100} \\ A \\ Z_{100} \end{pmatrix},$$

*where* $Z_{200} \in \mathbb{R}^{200\times 400}$ *and* $Z_{100} \in \mathbb{R}^{100\times 400}$ *are matrices consisting of all zeros. The right-hand sides* $\hat{b}^\delta$ *and* $\tilde{b}^\delta$ *are augmented similarly with zero elements.*

*We apply LSQR to the original system, as well as GMRES and RRGMRES to the two square systems, and we select the solutions with the overall smallest relative error compared to the true solution. Figure 4.23 shows contour plots of these solutions, as well as the corresponding relative errors and number of iterations. We note that GMRES and RRGMRES never converge and that the solutions are only nonzero in the parts of the solution domain corresponding to the parts of* $\widehat{A}$ *and* $\widetilde{A}$ *that are not zero-padded. Because these areas do not necessarily correspond to areas where the true solution is different from zero, we in fact construct solutions that do not have much in common with the wanted solution. The LSQR solution is seen to give contributions all over the domain and correspond somewhat to the true solution, even though the regularized solution is bad.*

Obviously, GMRES and RRGMRES are in general useless for solving augmented nonsquare systems, unless the sought solution has very special properties. We note also that the LSQR solution is not optimal, and indeed standard-form regularization is not optimal for reconstructing a smooth surface. We return to the above example in Chapter 6 where the problem is solved in general-form.

### 4.3.5 Inner Regularization

Regularizing CGLS and LSQR iterations applied to a discrete ill-posed problem need to be terminated before the final convergence is achieved due to the semi-convergent behavior. The solution quality generally degrades fast after the optimum is

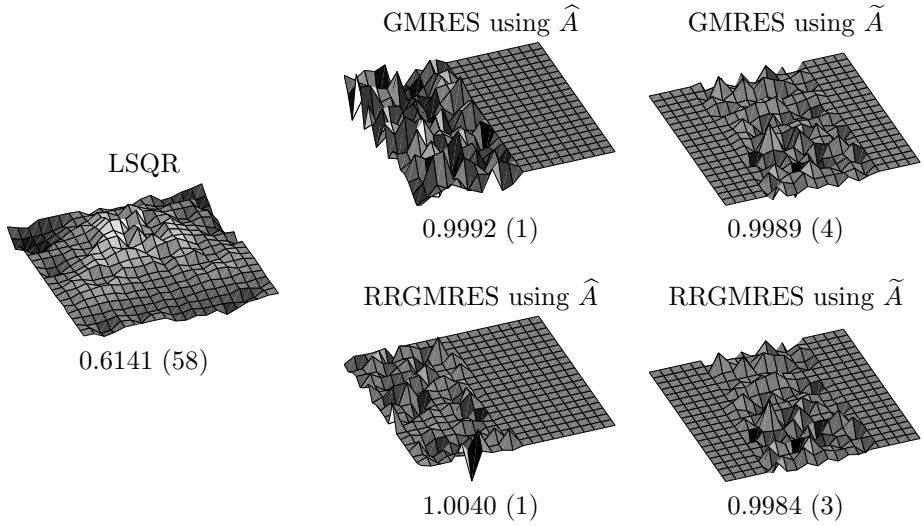



Figure 4.23: LSQR, GMRES and RRGMRES solutions with minimum relative error. The relative errors and the number of iterations are indicated below each plot.

achieved because the partial decomposition by the Lanczos method (4.23) starts to approximate the noise contaminated singular components. Moreover, the bidiagonal matrix $B_k$ from (4.23) inherits the ill-conditioning of $A$, and therefore the inner least squares problem $\min \|\beta e_1 - B_k y\|_2$ from (4.24) become ill-conditioned when too many iterations are performed.

To avoid degradation of the solution after the optimum is found, the concept of hybrid methods and inner regularization are found in the literature, see e.g., [33, 59] and reference therein. Basically, a regularized solution to the projected problem is sought, and direct methods can often be applied due to the small dimensions of the projected coefficient matrix $B_k \in \mathbb{R}^{(k+1)\times k}$. A regularized solution to the projected problem can, e.g., be computed by TSVD or Tikhonov regularization.

Kilmer and O'Leary [59] formulate regularized solutions of the projected problem in terms of the SVD of $B_k$. Let $B_k = Z_k \Gamma_k Q_k^T$ be the SVD of $B_k$, then the projected LSQR solution $\xi_k$ from (4.24) can be formulated as $\xi_k = (B_k^T B_k)^{-1} B_k^T \rho e_1 = Q_k \Gamma_k^\dagger Z_k^T \rho e_1$. Let $\overline{W}_k$ and $\overline{U}_{k+1}$ be the orthogonal matrices from (4.23) with columns that span the Krylov subspaces $\mathcal{K}(A^T A, A^T b)$ and $\mathcal{K}(AA^T, b)$, respectively, and remember that $\rho e_1 = \overline{U}_{k+1}^T b$. Then we can write the LSQR iterates as

$$x^{(k)} = \overline{W}_k Q_k \Phi_k \Gamma_k^\dagger Z_k^T \overline{U}_{k+1}^T b,$$

where the filter factors $\Phi_k = I$ are all one. The resemblance to the filtered SVD solution (2.13) is obvious, and both TSVD and Tikhonov regularization are easily applied to the projected problem by selecting the filter factors $\Phi_k$ according to (2.14).

The correspondence between direct regularization of the original problem, and projection followed by regularization of the projected problem basically relies on how well the singular values of $B_k$ in $\Gamma_k$ approximate the singular values of $A$, and how well the columns of $\overline{W}_k Q_k$ and $\overline{U}_{k+1} Z_k$ approximate the right and left singular vectors of $A$. (The convergence the Lanczos method is often connected to the so-called Rayleigh-Ritz approximations that approximate the eigenvalues and eigenvectors of a symmetric matrix when an OR method is applied, see e.g, [23, §5.2], [37, §6.3.2], and [80, §13].) Hanke and Hansen [33, §7.2] study LSQR with inner Tikhonov regularization, and theorems regarding both TSVD and Tikhonov regularization are found in [59, §3.5].

Traditionally, the Lanczos bidiagonalization algorithm has been used to provide the basis for the hybrid methods as described above. Nevertheless, similar relations can be obtained for other Krylov subspace methods, and it is interesting to study GMRES and MINRES as hybrid methods.

Let us follow an approach similar to the above for LSQR as indicated, e.g., in [52, 66], and define the SVD of the Hessenberg matrix $\overline{H}_k$ from the Arnoldi relation (4.13). I.e., let $\overline{H}_k = \widehat{Z}_k \widehat{\Gamma}_k \widehat{Q}_k$ be the SVD of $\overline{H}_k$, and write the projected MR solution (4.15) as $\hat{\xi}_k = (\overline{H}_k^T \overline{H}_k)^{-1} \overline{H}_k^T \rho e_1 = \widehat{Q}_k \widehat{\Gamma}_k^\dagger \widehat{Z}_k^T \rho e_1$. Now let the columns of $W_k$ from (4.12)–(4.13) span the Krylov subspace $\mathcal{K}(A, b)$ and note that $\rho e_1 = W_{k+1}^T b$. Then the GMRES iterates can be expressed by

$$\hat{x}^{(k)} = W_k \widehat{Q}_k \widehat{\Phi}_k \widehat{\Gamma}_k^\dagger \widehat{Z}_k^T W_{k+1}^T b.$$

where the filter factors $\widehat{\Phi}_k = I$ are chosen to be all one. Again, also the TSVD and Tikhonov filter factors (2.14) can in principle be chosen. But there is a severe problem with this formulation; the same basis $W_k$ (and $W_{k+1}$) is used on both left and right side because Arnoldi's method is based only on $A$ and not $A^T$. This implies that the singular values of $\overline{H}_k$ and the columns of $W_k \widehat{Q}_k$ can be thought of as approximations to the singular values and right singular vectors of $A$ – but the columns of $W_{k+1} Z_k$ are not approximations to the left singular vectors, see [66, §3]. Moreover, the convergence of GMRES is not tied to the Ritz values, but to the *harmonic Ritz values*, see, e.g., [27]. And while both the Ritz values and the harmonic Ritz values converge to the eigenvalues for well-conditioned problems, ill-conditioning of the Hessenberg matrix $\overline{H}_k$ means that the difference between the Ritz and harmonic Ritz values can be large, see [27, §4]. Therefore, inner regularization of GMRES might give unexpected results.

**Example 4.15** *We consider again the inverse heat problem from Example 4.8. Now the noise level is* $\|e\|_2/\|b\|_2 = 10^{-3}$ *such that also range components are affected by the noise. The problem therefore acts like a discrete ill-posed problem, and we need to compute regularized solutions. We apply LSQR and GMRES, as well as LSQR and GMRES with inner regularization. For the inner problem we use direct Tikhonov regularization and the regularization parameter is found by means of GCV, see Section 2.4. Figure 4.24 reports the relative errors compared to the true solution, and Fig. 4.25 shows the optimal solution iterates.*

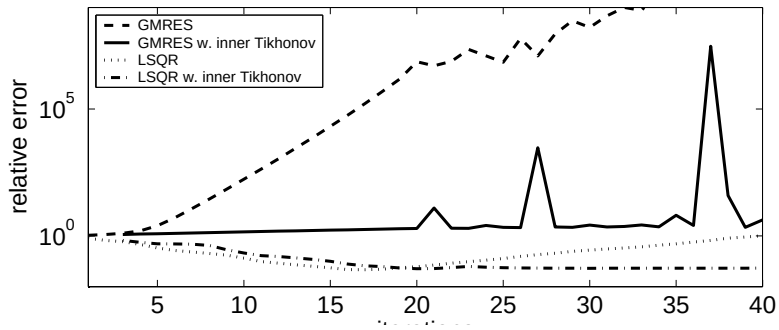



Figure 4.24: Relative errors for the iterates of LSQR and GMRES applied to the inverse heat problem with and without inner Tikhonov regularization.

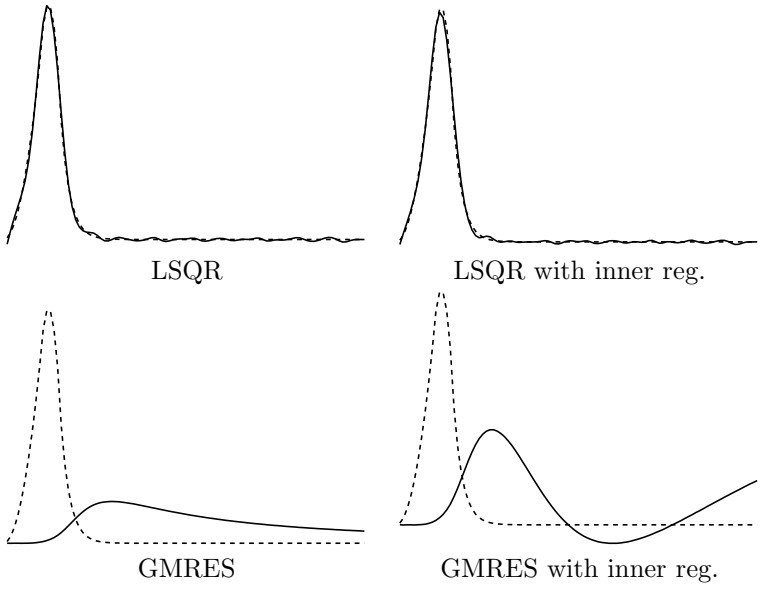



Figure 4.25: Optimal LSQR and GMRES solution iterates to the inverse heat problem with and without inner Tikhonov regularization where the regularization parameter is computed by means of GCV.

*We note that LSQR by itself again exhibits semi-convergence, i.e., the solution quality deteriorates fast after the optimum is achieved. By applying regularization to the projected problem, we note that the relative error stays small, also after the optimum is achieved. At this point, the Krylov subspace spans all the relevant SVD*

*components, and all further extensions to the Krylov subspace are filtered away by the inner Tikhonov regularization. The need for a stopping criterion is not as urgent, and in fact we might use the regularization parameter from the inner problem to determine when to stop the outer iterations.*

*For GMRES the situation is completely different. Firstly, GMRES by itself exhibits no semi-convergence. Secondly, while the inner Tikhonov regularization indeed has an effect compared to the standard GMRES iterates, it does not result in regularized solutions.*

## 4.4 Summary

In this chapter we introduced Krylov subspace methods as a class of general projection methods, and we described the basic differences between orthogonal-residual methods and minimum-residual methods. The main part of the chapter was concerned with analyses and numerical examples of these Krylov subspace methods when applied to discrete ill-posed problems. The analyses included a study of the minimization properties of the methods as well as studies of the underlying Krylov subspaces which were seen to be important for the regularizing effect.

To conclude the chapter, we note that all the results regarding CGLS/LSQR as well as MINRES and MR-II are in agreement with other similar results [31, 56, 58]. The results presented here characterize MINRES and MR-II as spectral filtering methods in the SVD basis and illustrate the minimization properties of the residual polynomials with intuitive examples. On the contrary, the results provided here for GMRES and RRGMRES do not, for general problems, completely agree with earlier studies of these methods [6, 10, 11]. The analyses and numerical examples clearly illustrate that neither GMRES nor RRGMRES can be characterized as general regularization methods even though both methods may provide regularized solutions for specific discrete ill-posed problems.

Chapter 5

# Discrete Smoothing Norms

When solving inverse problems we want to compute the most probable and physically correct solution given some measured data and a mathematical model. As mentioned in Chapter 2 it is sometimes favorable to solve the general-form Tikhonov problem instead of the standard-form problem. In this section, we will study different generalizations of regularization matrices to higher dimensions.

First, we look at a linear system $Ax = b$ and consider all the components in $x$ that belong to $\mathcal{N}(A)$ – either a true nullspace of $A$ (resulting from a nullspace of the underlying continuous operator $\mathcal{K}$), or a numerical nullspace connected to singular values at the machine precision. These components will naturally not be represented in $b$. Therefore, when reconstructing $x$ from $b$ by reversing the effect of $A$, there is no hope that we can reconstruct these components without additional information about the sought solution.

Both the TSVD solutions (2.9) and the standard-form Tikhonov solutions (2.12) are, from a mathematical point of view, based on a very natural choice if no additional information is available: compute solutions that avoid components from $\mathcal{N}(A)$. It is important to note that even though this philosophy seems reasonable, the computed solutions may have no physical meaning if important solution components indeed lie in $\mathcal{N}(A)$. Without further information, these components are effectively lost.

## 5.1 Regularization Matrices – One Dimension

Already Tikhonov [89] considered a more general formulation of the standard regularization problem. We consider only the discrete case, and only the case where the

2-norm is used both for the residual and the regularization term

$$x_{L,\lambda} = \operatorname{argmin}_x \|b - Ax\|_2^2 + \lambda^2 \|Lx\|_2^2 .$$

In principle, any matrix $L \in \mathbb{R}^{p\times n}$ can be used with the above formulation, even matrices with nontrivial nullspaces such that the regularization term becomes a seminorm. But for the minimizer to be unique, we need to require that $\mathcal{N}(A)\cap\mathcal{N}(L) = \{0\}$ [22, §9.2]. In vague terms, a solution component in a common nullspace of $A$ and $L$ will affect neither the residual term nor the regularization term, and thus a unique stabilized solution cannot be achieved.

As for the standard-form problem, we can also formulate the general-form problem in two equivalent ways

$$\min \left\| \begin{pmatrix} A \\ \lambda L \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2 \quad \text{and} \quad \left(A^T A + \lambda^2 L^T L\right) x = A^T b. \tag{5.1}$$

To solve the Tikhonov problem we need either to explicitly compute the GSVD and construct the solution as described in (2.16), or we can apply a least squares solver to the first expression above using the coefficient matrix $(A^T\,,\,\lambda L^T)^T$. The first approach is cumbersome for larger systems, but allows for an easy selection of the regularization parameter $\lambda$ when the GSVD has been computed. For the latter to be efficient, multiplication with $L$ and $L^T$, and of course also $A$ and $A^T$, must be not too expensive. Furthermore, the regularization parameter $\lambda$ must be chosen a priori. A third possibility is to apply iterative regularization to implicitly solve a system connected to the general-form problem which is the topic of the next chapter. It is mentioned here because this approach implies that it must be easy to compute minimum-norm least squares solutions of systems $Lx = y$ and $L^T y = x$. Furthermore, the approach needs to have available an explicit representation of the nullspace.

At this point it is also important to note that when considering only 2-norms, then any regularization matrix $L$ can always be substituted by an orthogonal transformation of $L$. I.e., obviously $\|Lx\|_2 = \|QRx\|_2 = \|Rx\|_2$ where $L = QR$ is a thin QR factorization of $L$, and $R$ is either triangular or trapezoidal.

Standard-form regularization results from the choice $L = I$. But other commonly used regularization matrices for one-dimensional problems are either diagonal weighting, or discrete approximations to derivative operators. For instance,

$$W_n = \begin{pmatrix} w_1 & & & \\ & w_2 & & \\ & & \ddots & \\ & & & w_n \end{pmatrix} \in \mathbb{R}^{n\times n} \tag{5.2}$$

$$L_n^{\{1\}} = \begin{pmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix} \in \mathbb{R}^{(n-1)\times n} \tag{5.3}$$

$$L_n^{\{2\}} = \begin{pmatrix} 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \end{pmatrix} \in \mathbb{R}^{(n-2)\times n}, \tag{5.4}$$

where $W_n$ is diagonal, $L_n^{\{1\}}$ defines a first order finite difference approximation to the first derivative of a signal of length $n$, and $L_n^{\{2\}}$ is a similar central finite difference approximation to the second derivative. It is assumed that the underlying gridpoints are equidistantly spaced, and actually that the grid spacing is one. Any equidistant grid spacing different from one results in a scaling of $L_n^{\{1\}}$ and $L_n^{\{2\}}$, but often this scaling is absorbed into the regularization parameter. Using $L_n^{\{1\}}$ or $L_n^{\{2\}}$ as regularization matrices restrict the gradient or the curvature of the solution instead of the size. That is, we seek solutions with a certain "flatness" or "smoothness." We note that in the following, a general regularization matrix is denoted $L$, whereas a specific approximation to a derivative operator has subscript(s) denoting the size, and superscript(s) denoting the type of derivative. The nullspaces of $L_n^{\{1\}}$ and $L_n^{\{2\}}$ are both low-dimensional and smooth, i.e., they are given as

$$\begin{aligned} \mathcal{N}(L_n^{\{1\}}) &= \operatorname{span}\left\{ \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix}^T \right\} \\ \mathcal{N}(L_n^{\{2\}}) &= \operatorname{span}\left\{ \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix}^T, \begin{pmatrix} 1 & 2 & \dots & n \end{pmatrix}^T \right\}. \end{aligned}$$

Due to the inherent smoothing property of the coefficient matrix $A$, the nullspace of $A$ will generally be high-frequent and therefore not intersect with the nullspaces of $L_n^{\{1\}}$ and $L_n^{\{2\}}$.

The following example shows how general-form regularization can be more appropriate than regularization on standard form.

**Example 5.1** *Consider a discretization by means of Gauss-Laguerre quadrature of the inverse Laplace transformation given by the kernel* $K(s,t) = e^{-st}$ *and with integration intervals* $[0\,;\infty[$. *We use the implementation* ilaplace *from MOORe Tools to obtain the discretized kernel* $A \in \mathbb{R}^{n\times n}$. *The continuous inverse Laplace operator has no nullspace, but the discrete coefficient matrix A is severely ill-conditioned. For a discretization using* $n = 100$, *the numerical rank of A (the number of singular values above machine precision) is approximately 30, i.e., the numerical range of* $A^T$ *is* $\mathcal{R}(A^T) \approx \operatorname{span}\{v_1, \dots, v_{30}\}$. *These vectors mainly live in the left part of the domain, i.e.,* $v_i(t) \approx 0$, $t \geq t_0$ *for* $i = 1, \dots, 30$ *and any solution component in the right part of the domain effectively lies in the numerical nullspace of A,* $\mathcal{N}(A)$. *We look at two different true solutions*

$$f(t) = e^{-\frac{t}{2}} \quad , \quad \bar{f}(t) = 1 - e^{-\frac{t}{2}},$$

*where the first decays exponentially for* $t \to \infty$, *and the second approaches the constant one for* $t \to \infty$. *Discrete representations* $x$ *and* $\bar{x}$ *of the two cases are shown in Fig. 5.1 (top). Figure 5.1 (bottom) shows* $x$ *and* $\bar{x}$ *expressed in the right singular vectors* $|V^T x|$ *(solid lines). These plots show that* $x$ *is quite well represented in*

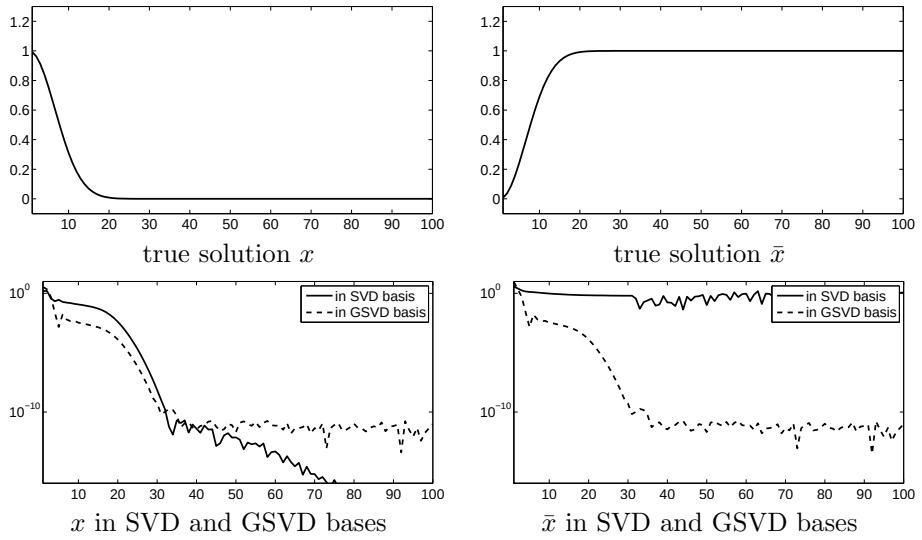



Figure 5.1: True solutions for the two cases of the ilaplace test problem. Also shown are the true solutions expressed in the basis of the right singular vectors, as well as the true solutions in the GSVD basis.

span $\{v_1, \ldots, v_{30}\}$, *whereas this is not the case for* $\bar{x}$. *This indicates that standard-form regularization will most likely be able to reconstruct* $x$, *but probably not* $\bar{x}$.

*The standard-form Tikhonov solutions with minimum relative error are shown in Fig. 5.2 (solid lines). For* $x$ *it makes sense to avoid components in* $\mathcal{N}(A)$ *because* $x$ *contains no significant components here anyway. For* $\bar{x}$ *it does not make sense to disregard components in* $\mathcal{N}(A)$. *Now we exchange the regularization term* $\|x\|_2$ *with* $\|L_n^{\{1\}}x\|_2$, *and intend to produce "flat" solutions, i.e., "anything we do not know about – make it flat." The results are shown by dash-dotted lines in Fig. 5.2. Obviously, we are now able to choose sensible information from* $\mathcal{N}(A)$ *for reconstructing* $\bar{x}$. *It is interesting to note that using the smoothing norm* $\|L_n^{\{1\}}x\|_2$ *of course also makes perfect sense for reconstructing* $x$ *because in addition to approximating zero in the right part of the domain, the solution is definitely also "flat." Note also from Fig. 5.1 (bottom) that both* $x$ *and* $\bar{x}$ *are well represented in the GSVD basis based on the matrix pair* $(A, L_n^{\{1\}})$.

### 5.1.1 Extended Regularization Matrices

The regularization matrices described above are the standard cases, and of course we might consider also more general regularization matrices. In this section we consider a few generalizations.

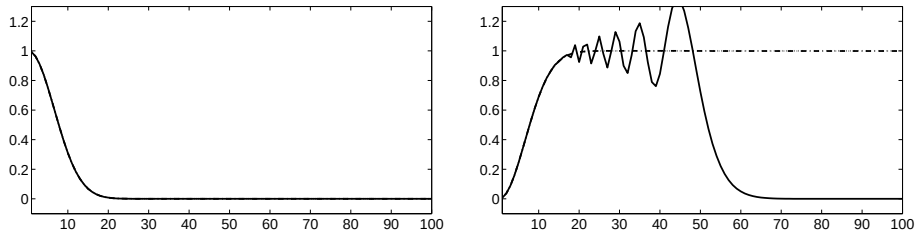



Figure 5.2: Solutions to (solid lines) standard-form Tikhonov problem, and (dash dotted lines) general-form Tikhonov with smoothing-norm $L_1$. A dotted line shows the true solutions (coincides with the general-form solutions.)

### Boundary Conditions

The standard derivative matrices (5.3)–(5.4) are rectangular with more columns than rows, and the derivatives are not computed at the boundaries. As was the case for PSFs for one- and two-dimensional image deblurring, we can consider a variety of boundary conditions. By using boundary conditions, we again impose additional information onto the solution, and the eventual choice of boundary conditions should reflect the nature of the wanted solution.

Consider the discrete one-dimensional signal $x \in \mathbb{R}^n$, and let $P \in \mathbb{R}^{n \times n}$ be the reversal matrix. Then the most general boundary conditions zero, periodic, and reflexive, result in the extended signals

$$x^{\{z\}} = \begin{pmatrix} 0 & x^T & 0 \end{pmatrix}^T \tag{5.5}$$

$$x^{\{p\}} = \begin{pmatrix} x^T & x^T & x^T \end{pmatrix}^T \tag{5.6}$$

$$x^{\{r\}} = \begin{pmatrix} (Px)^T & x^T & (Px)^T \end{pmatrix}^T. \tag{5.7}$$

The $L$ matrices that implement boundary conditions become square, but not necessarily of full rank. The approximation to the second derivative operator that implements zero boundary conditions is denoted by $L_n^{\{2z\}}$ and takes the form

$$L_n^{\{2z\}} = \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix} \in \mathbb{R}^{n \times n}. \tag{5.8}$$

It has Toeplitz structure and full rank. The $L$ matrices that implement approximations to second derivatives operators with periodic and reflexive boundary conditions are similarly denoted $L_n^{\{2p\}}$ and $L_n^{\{2r\}}$. These are also square, but have rank $n-1$ with the nullspaces spanned by a constant vector. These matrices are circulant and Toeplitz-plus-Hankel, respectively.

**Sum of Norms**

We can also use the general regularization matrices above to generate approximations to Sobolev norms or combined norms. E.g., we can formulate the regularization term $\|x\|_2 + \|L_n^{\{1\}}x\|_2$ which can be written as $\|L_n^{\{\text{Sob}\}}x\|_2$ where

$$L_n^{\{\text{Sob}\}} = \begin{pmatrix} I_n \\ L_n^{\{1\}} \end{pmatrix} \tag{5.9}$$

is a "stacked" operator consisting of an identity and an approximation to the first derivative operator.

Note that the boundary conditions considered above can also be formulated as a sum of norms. If $B \in \mathbb{R}^{(n-p)\times n}$ is a rectangular matrix that contains a few rows that define the boundary conditions, then the regularization terms can be formulated as $\|Lx\|_2 + \|Bx\|_2$. E.g., for the second derivative using periodic boundary conditions, we define

$$B = \begin{pmatrix} -2 & 1 & 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & 1 & -2 \end{pmatrix} \in \mathbb{R}^{2\times n},$$

and the regularization term is $\|L_n^{\{2p\}}\|_2 = \|L_n^{\{2\}}x\|_2 + \|Bx\|_2$.

## 5.2 Higher Dimensions

For problems of more than one dimension, the concept of derivatives is more complicated. We can consider derivatives along the axis dimensions seperately, or we can consider combined derivatives. Furthermore, we can consider different orders of derivatives in different directions.

Consider now two different uses of directional derivatives in more than one dimension. Either compute the norm of a sum of directional derivatives, or compute the sum of the norms of directional derivatives. I.e.,

$$\left\| \frac{\partial^p f}{\partial x_1^p} + \frac{\partial^p f}{\partial x_2^p} + \cdots + \frac{\partial^p f}{\partial x_N^p} \right\|_2^2 \quad \text{or} \quad \left\| \frac{\partial^p f}{\partial x_1^p} \right\|_2^2 + \left\| \frac{\partial^p f}{\partial x_2^p} \right\|_2^2 + \cdots + \left\| \frac{\partial^p f}{\partial x_N^p} \right\|_2^2,$$

where $N$ is the number of dimensions and $p$ is the order of the derivatives.

The second approach is comparable to the Sobolev norm described above, i.e., a sum of different regularization terms. The first formulation is more complicated because computing the sum of directional derivatives needs to be a linear operation that can be formulated by a single regularization matrix $L$.

We base the discussion on two-dimensional problems, e.g., image reconstruction. Images are two-dimensional functions, $f(s,t)$, and as such we can define the partial derivatives of the image with respect to the two cartesian coordinate axes $s$ and $t$. Measures connected to the Laplacian are often used in image restoration and two-dimensional smoothing [55, 71, 77], and if $p = 2$, we get

$$\nabla^2 f = \frac{\partial^2 f}{\partial s^2} + \frac{\partial^2 f}{\partial t^2},$$

and we can use a discrete approximation to the regularization term $\|\nabla^2 f\|_2$. Discrete approximations to the Laplacian are also often used to form variance filters in connection with image segmentation and edge detection, see, e.g., [51, §9.4].

The second approach, considering the directional derivatives separately, can be implemented using any combination of derivatives because the one-dimensional derivatives along each dimension are used independently of one another. But to be consistent with the Laplacian, most of the following studies are based on the second derivatives in both axis directions

$$d_s(s,t) = \frac{\partial^2 f}{\partial s^2} \qquad \text{and} \qquad d_t(s,t) = \frac{\partial^2 f}{\partial t^2}.$$

This approach uses as the regularization term a discrete version of $\|d_s\|_2^2 + \|d_t\|_2^2$. In the following, this approach is called the Sobolev approach.

Both measures will restrict the roughness of the solution, but as we shall see, the applicability of the two approaches differ.

### 5.2.1 Discrete Laplacian

We can look at the computation of the Laplacian of an image in two different ways, as a convolution of the image with a filter kernel or as a matrix-vector product. If we define the five-point stencil

$$F = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

then the convolution of the filter kernel $F$ with the discrete image $X \in \mathbb{R}^{m\times n}$, $D = F \circledast X$, yields approximations to $\nabla^2 f$ of each interior point of the image $X$. But to apply the stencil $F$ to the entire image, one has to take special care of the boundaries as the stencil cannot be directly applied to the boundary points.

If we address the problem from the matrix point of view then we can compute the discrete derivatives along each dimension as

$$D_x = X L_n^{\{2\}T}, \quad D_y = L_m^{\{2\}} X,$$

which can be formulated equivalently using Kronecker products as

$$d_x = (L_n^{\{2\}} \otimes I_m)x, \quad d_y = (I_n \otimes L_m^{\{2\}})x,$$

where $(L_n^{\{2\}} \otimes I_m) \in \mathbb{R}^{m(n-2)\times mn}$, $(I_n \otimes L_m^{\{2\}}) \in \mathbb{R}^{n(m-2)\times mn}$, and $x = \operatorname{vec}(X)$ is the columnwise stacked image. The convolution with the stencil $F$ can therefore equivalently be formulated as a matrix-vector multiplication where the matrix is the sum of the two Kronecker products. But obviously, the two Kronecker products are only of same size if $n = m$, i.e., if the considered image is square. And even in this case, corresponding elements in the two Kronecker products will not correspond the same pixels, and a direct sum of the two will not yield the point-wise sum of second

derivatives. That is, the lack of boundary conditions is here reflected in the mismatch of the Kronecker products.

To formulate the discrete Laplacian, we therefore need to consider some kind of boundary conditions. First, we look at zero boundary conditions, i.e., we extend the image with a border of zeros

$$X^{\{z\}} = \begin{bmatrix} 0 & 0_n^T & 0 \\ 0_m & X & 0_m \\ 0 & 0_n^T & 0 \end{bmatrix},$$

where $0_n$ is a zero vector of length $n$. These boundary conditions are also called Dirichlet conditions [74] because the pixel value outside the boundary is fixed. We can now compute the convolution $F \circledast X^{\{z\}}$, i.e., including the boundary points of the original image $X$. We can also express the convolution in matrix-vector notation $L_{mn}^{\{2z\}} x$ where

$$L_{mn}^{\{2z\}} = \begin{pmatrix} D & I_m & & & \\ I_m & D & I_m & & \\ & \ddots & \ddots & \ddots & \\ & & I_m & D & I_m \\ & & & I_m & D \end{pmatrix} \in \mathbb{R}^{mn \times mn}.$$

Here $I_m$ is the identity matrix of size $m \times m$ and the blocks $D$ on the diagonal are given by

$$D = \begin{pmatrix} -4 & 1 & & & \\ 1 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 1 \\ & & & 1 & -4 \end{pmatrix} \in \mathbb{R}^{m \times m}$$

The matrix $L_{mn}^{\{2z\}}$ is the result of applying the one-dimensional second derivative operator with zero boundary conditions in the two directions, i.e.,

$$L_{mn}^{\{2z\}} = \left(L_n^{\{2z\}} \otimes I_m\right) + \left(I_n \otimes L_m^{\{2z\}}\right).$$

Note that here $L_n^{\{2z\}}$ and $L_m^{\{2z\}}$ are square matrices defined in (5.8), and that the sizes of the Kronecker products are identical. Obviously, $L_{mn}^{\{2z\}}$ is block Toeplitz with Toeplitz blocks.

Using the reversal matrices $P_n \in \mathbb{R}^{n \times n}$ and $P_m \in \mathbb{R}^{m \times m}$ then the periodic and reflexive extensions to the image can be identified as

$$X^{\{p\}} = \begin{bmatrix} X & X & X \\ X & X & X \\ X & X & X \end{bmatrix}, \quad X^{\{r\}} = \begin{bmatrix} P_m^T X P_n & P_m^T X & P_m^T X P_n \\ X P_n & X & X P_n \\ P_m^T X P_n & P_m^T X & P_m^T X P_n \end{bmatrix}.$$

The reflexive boundary conditions are sometimes called Neumann boundary conditions [74] because reflecting the image out of the image domain leads to a restriction of the first derivative across the boundary. A convolution of $X^{\{p\}}$ or $X^{\{r\}}$ with the stencil $F$ will again result in approximations of the second derivatives at the boundaries of $X$. In a matrix-vector notation the $L$ matrices can again be described as a sum of two Kronecker products. In case of periodic bondary conditions, $L_{mn}^{\{2p\}} = (L_n^{\{2p\}} \otimes I_m) + (I_n \otimes L_m^{\{2p\}})$ is block circulant with circulant blocks (BCCB) and it can be diagonalized by the 2D-FFT. In case of reflexive boundary conditions, the structure of $L_{mn}^{\{2r\}} = (L_n^{\{2r\}} \otimes I_m) + (I_n \otimes L_m^{\{2r\}})$ is block Toeplitz-plus-Hankel with Toeplitz-plus-Hankel blocks (BTHTHB), and can be diagonalized by the 2D-DCT because the one-dimensional operator is symmetric.

Instead of explicitly considering boundary conditions, one can also apply the stencil $F$ only to the inner points of the image, and then extrapolate values for the derivatives to the boundary points using the calculated inner points. This approach is e.g., implemented in the MATLAB function del2 where a simple linear extrapolation of the derivatives is used. Specifically, if $d_i = x_{i-1} - 2x_i + x_{i+1}$, for $i = 2, 3, \ldots, n-1$ denote the discrete approximations to the second derivatives in the points $x_i$, then the approximation to the directional derivatives on the boundaries are given as

$$\begin{aligned} d_1 &= 2d_2 - d_3 = 2x_1 - 5x_2 + 4x_3 - x_4 \\ d_n &= 2d_{n-1} - d_{n-2} = -x_{n-3} + 4x_{n-2} - 5x_{n-1} + 2x_n \ . \end{aligned}$$

These calculations are carried out for the two dimensions independently, and the resulting second derivative images are summed up. This amounts again to a representation by the sum of two Kronecker products, and the extrapolation approach results in the following modified one-dimensional second derivative operator

$$L_n^{\{2e\}} = \begin{pmatrix} 2 & -5 & 4 & -1 & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & -1 & 4 & -5 & 2 \end{pmatrix}.$$

This leads to $L_{mn}^{\{2e\}} = (L_n^{\{2e\}} \otimes I_m) + (I_n \otimes L_m^{\{2e\}})$, which takes the the more complicated form

$$L_{mn}^{\{2e\}} = \begin{pmatrix} C & -5I_m & 4I_m & -I_m & \\ I_m & \overline{D} & I_m & & \\ & \ddots & \ddots & \ddots & \\ & & I_m & \overline{D} & I_m \\ -I_m & 4I_m & -5I_m & C & \end{pmatrix} \in \mathbb{R}^{mn \times mn}$$

where

$$C = \begin{pmatrix} 4 & -5 & 4 & -1 & & \\ 1 & & 1 & & & \\ & \ddots & & \ddots & & \\ & & 1 & & & 1 \\ & -1 & 4 & -5 & 4 & \end{pmatrix} \quad \text{and} \quad \overline{D} = \begin{pmatrix} & -5 & 4 & -1 & & \\ 1 & -4 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & 1 & -4 & 1 \\ & -1 & 4 & -5 & & \end{pmatrix} \in \mathbb{R}^{m \times m} .$$

The resulting matrix is neither Toeplitz or Hankel nor symmetric, and therefore, it cannot be diagonalized by the 2D-FFT or the 2D-DCT. And as we shall see later, the nullspace of this operator is undesirable.

Above, we have applied boundary conditions to compute second derivative approximations also at the boundaries of the image, and we have seen that all the approaches can be described in matrix notation as a sum of two Kronecker products. From the matrix point of view, a more brutal possibility for making the Kronecker products fit each other is by truncating the two Kronecker products such that $L_{mn}^{\{2t\}} = (L_n^{\{2\}} \otimes \widehat{I}_m) + (\widehat{I}_n^T \otimes L_m^{\{2\}})$ where two rows have been removed in $\widehat{I}_m$, and $\widehat{I}_n$. Unfortunately, this approach results in disregarding the corresponding rows and columns of the image, and thereby enlarge the dimension of the nullspace.

#### 5.2.1.1 Nullspaces

It is important for regularization matrices to have low-dimensional nullspaces, and especially nullspaces that do not intersect with the nullspace of $A$. Furthermore, when using iterative regularization it is important to have a basis for the nullspace given explicitly.

Unfortunately, it is difficult to identify the nullspace of a general matrix of the form $(B_1 \otimes C_1) + (B_2 \otimes C_2)$. However, when the resulting matrix is structured BCCB, BTHTHB, etc., we can diagonalize it and thereby compute vectors that span the nullspace.

Specifically, the boundary conditions considered above when applied to the approximations to the discretized two-dimensional Laplacian gives the following nullspaces

$$\begin{aligned} \mathcal{N}(L_{mn}^{\{2z\}}) &= \text{span}\{0\} \\ \mathcal{N}(L_{mn}^{\{2p\}}) &= \text{span}\{(1, 1, \ldots, 1)^T\} \\ \mathcal{N}(L_{mn}^{\{2r\}}) &= \text{span}\{(1, 1, \ldots, 1)^T\} \\ \dim(\mathcal{N}(L_{mn}^{\{2e\}})) &= 8 \\ \dim(\mathcal{N}(L_{mn}^{\{2t\}})) &= 2m + 2n - 4 . \end{aligned} \tag{5.10}$$

When applying zero boundary conditions, the operator has no nullspace, and for periodic and reflexive boundary conditions the nullspace is one-dimensional and spanned by a constant function. In the last two cases the nullspaces are not as simple, and only the dimensions of the nullspaces are given. Figure 5.3 shows eight

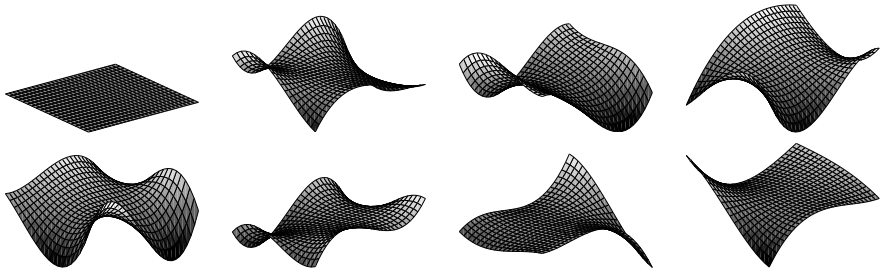


Figure 5.3: Two-dimensional illustration of eight orthonormal vectors that span the nullspace of the approximation to the two-dimensional Laplacian using the extrapolated approach for calculating the derivatives on the boundary.

orthonormal vectors that span $\mathcal{N}(L_{mn}^{\{2e\}})$ for $m = 20$ and $n = 30$. The vectors are shown as two-dimensional surfaces, and obviously the nullspace allows a great deal of variation in the solution. In the last case of (5.10), using a truncation of the identity matrices to make the Kronecker products fit each other, the dimension of the nullspace increases with the problem size. Obviously, the last two choices are probably not wise choices in connection with regularization of ill-posed problems.

#### 5.2.1.2 Computational Aspects

The formulation of the $L$ matrices and the behavior of the nullspaces as described above are important aspects. But it is also important that computations using the $L$ matrices are efficiently implemented. Assume that we want to compute the Tikhonov solution $x_\lambda$ for some value of $\lambda$. We can apply any least squares solver to the least squares problem with the coefficient matrix $(A^T, \lambda L^T)^T$, and due to the formulation through Kronecker products, or convolutions with the filter kernel $F$, all matrix-vector multiplications with $L$ and $L^T$ can be carried out efficiently. But for large-scale problems we might need a lot of iterations, and/or an efficient preconditioner. Furthermore, we need to choose the value $\lambda$ a priori, and repeat the solution process for every new regularization parameter.

If $A$ and $L$ have the same structure – BCCB, BTHTHB, etc. – then both matrices can be diagonalized simultaneously. Assume, e.g., that both $A$ and $L$ are formulated using reflexive boundary conditions, and that the PSF described by $A$ is spatially invariant and symmetric. Then $A = \Psi\Lambda_A\Psi^T$ and $L = \Psi\Lambda_L\Psi^T$ where $\Psi \in \mathbb{R}^{nm\times nm}$ is the 2D-DCT matrix. Using the formulation $(A^TA + \lambda^2 L^TL)x = A^Tb$, we get

$$(\Lambda_A^2 + \lambda^2\Lambda_L^2)\Psi x = \Lambda_A\Psi^T b,$$

and similar to (2.12), where the Tikhonov solution for the standard-form problem is formulated in terms of filter factors in the SVD basis, we can now write the Tikhonov solution using filter factors in the 2D-DCT basis. This approach is e.g., used in the

MATLAB toolbox Restore Tools by Nagy et al. [71] for computing Tikhonov solutions to image deblurring problems.

Finally, we can also solve a problem connected to the general-form formulation by means of iterative regularization as we shall see in the next chapter. To do this we need also to be able to solve systems with $L$ and $L^T$, i.e., we need at least implicitly to work with the Moore-Penrose pseudoinverse of $L$ and $L^T$. In case $L$ is easily diagonizable through the 2D-FFT, 2D-DCT, etc., it is also easy to compute the pseudoinverse. I.e., for proper choices of the boundary conditions, this can be accomplished for the stencil approach. On the other hand, a general formulation through a sum of two Kronecker products is not easily inverted and thus the $L$ matrix arising from extrapolated boundary conditions is not easily applicable in connection with iterative regularization.

#### 5.2.1.3 Generalizations

The approach can to some extend be generalized to higher dimensions to implement approximations to the $N$-dimensional Laplacian. Care should be taken if other matrices than full-rank or second derivative approximations with proper boundaries are used because the nullspace is not necessarily well behaved.

### 5.2.2 Sobolev Approach

We now turn to the case where the directional derivatives are kept separated, and we start again by considering the two-dimensional case and second derivatives. Using the standard discrete derivative operators, we define the derivatives in the two directions as

$$D_x = XL_n^{\{2\}T}, \quad D_y = L_m^{\{2\}}X.$$

which can be reformulated for vectorized images using Kronecker products as

$$d_x = (L_n^{\{2\}} \otimes I_m)x, \quad d_y = (I_n \otimes L_m^{\{2\}})x.$$

Now, instead of adding the two Kronecker products together, we let $\|d_x\|_2^2 + \|d_y\|_2^2$ be a measure of the size of the solution, and this can equivalently be formulated as $\|L_{mn}^{\{s\}}x\|_2^2$ where

$$L_{mn}^{\{s\}} = \begin{pmatrix} L_n^{\{2\}} \otimes I_m \\ I_n \otimes L_m^{\{2\}} \end{pmatrix}, \tag{5.11}$$

i.e., as a Sobolev norm. In this framework, the size of the two operators need not be identical, and we can use the original rectangular $L$ matrices without considering boundary conditions. We can also use the same boundary conditions as above, or even different boundary conditions in the two directions, e.g., in (5.11) substitute $L_n^{\{2z\}}$ for $L_n^{\{2\}}$, and $L_m^{\{2p\}}$ for $L_m^{\{2\}}$.

This formulation is easily generalized to include other derivatives, diagonal weighting matrices, and even the standard 2-norm of the solution which is obtained by adding the row $I_n \otimes I_m$ to $L_{mn}^{\{s\}}$. The stack of Kronecker products is also easily extended to higher dimensions, i.e., for an $N$-dimensional problem the block rows will consist of $N$-term Kronecker products. We therefore define the general stacked Kronecker array

$$L_N^{\{s\}} = \begin{pmatrix} Z_{d_N}^{\{1\}} \otimes Z_{d_{N-1}}^{\{1\}} \otimes \cdots \otimes Z_{d_1}^{\{1\}} \\ Z_{d_N}^{\{2\}} \otimes Z_{d_{N-1}}^{\{2\}} \otimes \cdots \otimes Z_{d_1}^{\{2\}} \\ \vdots \\ Z_{d_N}^{\{R\}} \otimes Z_{d_{N-1}}^{\{R\}} \otimes \cdots \otimes Z_{d_1}^{\{R\}} \end{pmatrix}, \tag{5.12}$$

where $R \in \mathbb{N}$ is the number of block rows (number of terms in the Sobolev norm) and $Z_{d_i}^{\{j\}}$ is the matrix at the $j$th block row of $L_N^{\{s\}}$ corresponding to the $i$th dimension of size $d_i$. Each $Z_{d_i}^{\{j\}}$ can be any discrete derivative operator, diagonal weighting matrix or any other regularization operator that should be applied to the $i$th dimension, even just the identity. But again, operations with the resulting matrix $L_N^{\{s\}}$ should be easily implementable, the nullspace should be well behaved, and for applying iterative regularization methods, also the vectors $L_N^{\{s\}\dagger}x$ and $L_N^{\{s\}\dagger T}y$ should be easy to compute.

#### 5.2.2.1 Nullspace

The nullspaces for these regularization matrices are somewhat easier to investigate than the nullspaces for the stencil approach. Consider first the two-dimensional approach using second derivative approximations. A nullspace vector is characterized by

$$\|d_x\|_2 + \|d_y\|_2 = 0 \quad \Leftrightarrow \quad \|d_x\|_2 = 0 \wedge \|d_y\|_2 = 0.$$

i.e., the nullspace of $L_{mn}^{\{s\}}$ from (5.11) is the common nullspace of the nullspaces of both rows of $L_{mn}^{\{s\}}$. We state the following.

**Lemma 5.1** *The nullspace of the two-dimensional stacked Kronecker array $L_{mn}^{\{s\}}$ from* (5.11) *is given by* $\mathcal{N}(L_{mn}^{\{s\}}) = \operatorname{span}\left\{(w \otimes \bar{w}) \,|\, w \in \mathcal{N}(L_n^{\{2\}}) \wedge \bar{w} \in \mathcal{N}(L_m^{\{2\}})\right\}$.

Proof. First note from the Kronecker SVD in Lemma 3.7 that all singular vectors of a Kronecker products can be written as a Kronecker product of two smaller vectors. Therefore, all vectors that span the nullspace of a Kronecker product matrix must also be a Kronecker product. Now let a nullspace vector be given as $w \otimes \bar{w}$, and we get

$$(L_n^{\{2\}} \otimes I_m)(w \otimes \bar{w}) = (L_n^{\{2\}} w \otimes I_m \bar{w}) = 0 \Leftrightarrow L_n^{\{2\}} w = 0 \vee I_m \bar{w} = 0$$

Trivially $I_m \bar{w} = 0 \Leftrightarrow \bar{w} = 0$, and the above expression holds for $L_n^{\{2\}} w_i = 0 \vee \bar{w} = 0$, i.e., the nontrivial nullspace vectors are obtained for $w \in \mathcal{N}(L_n^{\{2\}}) \wedge \bar{w} \in \mathbb{R}^m$. A similar conclusion holds for the second block row of $L_{mn}^{\{s\}}$, i.e., $(I_n \otimes L_m^{\{2\}})(w \otimes \bar{w}) = 0 \Leftrightarrow \bar{w} \in \mathcal{N}(L_m^{\{2\}}) \wedge w \in \mathbb{R}^n$. Thus it must hold that both $w \in \mathcal{N}(L_n^{\{2\}})$ and $\bar{w} \in \mathcal{N}(L_m^{\{2\}})$ which concludes the proof. □

We continue with the nullspace of the general stacked Kronecker array (5.12). In this case the nullspace is not quite as easy to determine. We state the following lemma.

**Lemma 5.2** *The nullspace of* $L_N^{\{s\}}$ *from* (5.12) *is given by*

$$\mathcal{N}(L_N^{\{s\}}) = \operatorname{span}\left\{ (w_{d_N} \otimes w_{d_{N-1}} \otimes \cdots \otimes w_{d_1}) \,|\, w_{d_i}\text{s defined below} \right\}.$$

*For all block rows* $k = 1, \ldots, R$ *of* $L_N^{\{s\}}$ *and at least one* $i \in [\,1, \ldots, N\,]$*, it must hold that* $Z_{d_i}^{\{k\}} w_{d_i} = 0$*. Moreover, all other* $w_{d_j} \in \mathbb{R}^{d_j}$*,* $j \in [\,1, \ldots, N\,]$*,* $j \neq i$ *can be chosen arbitrarily for this block row. I.e., for each block row* $k$*, it holds that*

$$w_{d_i} \in \mathcal{N}(Z_{d_i}^{\{k\}}) \wedge w_{d_j} \in \mathbb{R}^{d_j} \quad \text{for} \quad i, j \in [\,1, \ldots, N\,] \quad , \quad j \neq i.$$

*If any block row has no nontrivial nullspace, then* $L^{\{s\}}$ *has no nontrivial nullspace.*

Proof. First note that if $L_N^{\{s\}} w = 0$ then the same must hold for every individual block row, i.e., $(Z_{d_n}^{\{k\}} \otimes \cdots \otimes Z_{d_1}^{\{k\}}) w = 0$, for $k = 1, \ldots, R$. Conversely, if any block row does not have a nontrivial nullspace, then the $L_N^{\{s\}}$ has no nontrivial nullspace. Using Lemma 3.7 (Kronecker SVD), we note that all nullspace vectors $w$ can be expressed as $N$-term Kronecker products. Therefore we can write the nullspace vectors as $w = (w_{d_n} \otimes \cdots \otimes w_{d_1})$ for some vectors $w_{d_i} \in \mathbb{R}^{d_i}$ for $i = 1, \ldots, N$. For each block row, we get $(Z_{d_n}^{\{k\}} w_{d_N} \otimes \cdots \otimes Z_{d_1}^{\{k\}} w_{d_1}) = 0 \Leftrightarrow Z_{d_n}^{\{k\}} w_{d_N} = 0 \vee \cdots \vee Z_{d_1}^{\{k\}} w_{d_1} = 0$. Therefore for some $i \in [\,1, \ldots, N\,]$ it must hold that $w_{d_i} \in \mathcal{N}(Z_{d_i}^{\{k\}})$. Moreover, the remaining $w_{d_j}$, $j \in [\,1, \ldots, N\,]$, $j \neq i$ can be chosen arbitrarily. □

It is clear from Lemma 5.2 that not all combinations of derivative operators lead to regularization matrices with low-dimensional and smooth nullspaces.

**Example 5.2** *Consider the stacked regularization matrix*

$$L_{nmp}^{\{s\}} = \begin{pmatrix} L_n^{\{1\}} \otimes I_m \otimes I_p \\ I_n \otimes L_m^{\{1\}} \otimes I_p \end{pmatrix},$$

*that implements first derivative smoothing of the first and second dimension. Due to Lemma 5.2 the nullspace is given by the vectors* $w = (\, w_n \otimes w_m \otimes w_p \,)$ *where* $w_n \in \mathcal{N}(L_n^{\{1\}})$*,* $w_m \in \mathcal{N}(L_m^{\{1\}})$*, and* $w_p \in \mathbb{R}^p$*. That is,* $\mathcal{N}(L_{nmp}^{\{s\}})$ *is not restricted in the third dimension in which the nullspace vectors are allowed to be arbitrary.*

The example above illustrates that despite the very general formulation of stacked regularization matrices (5.12), care must be taken when using general stacked Kronecker arrays as regularization matrices. In the following, we look at the computational aspects of the stacked Kronecker matrices.

#### 5.2.2.2 Computational Aspects

The $L$ matrices will typically consist of more than one block row. In case zero, periodic, or reflexive boundary conditions are used, then clever diagonalization schemes can again be used to diagonalize each block row. E.g., consider the two-dimensional case using the stacked Kronecker products from (5.11), and consider reflexive boundary conditions for both $A$, $L_m^{\{2\}}$ and $L_n^{\{2\}}$. Then all three can be simultaneously diagonalized by the 2D-DCT matrix. Let $A = \Psi\Lambda_A\Psi^T$, $(L_n^{\{2\}} \otimes I_m) = \Psi\Lambda_1\Psi^T$, and $(I_n \otimes L_n^{\{2\}}) = \Psi\Lambda_2\Psi^T$ where $\Psi \in \mathbb{R}^{mn\times mn}$ is the 2D-DCT matrix, then the Tikhonov problem can be formulated as

$$\left(\Lambda_A + \lambda^2(\Lambda_1^2 + \Lambda_2^2)\right)\Psi^T x = \Lambda_A\Psi^T b,$$

i.e., the solutions can again be formulated in terms of 2D-DCT filter factors.

But there are special cases for which clever factorization schemes can be used to factorize more general stacked Kronecker arrays. These factorizations can then in turn be used to perform fast matrix-vector multiplications and fast system solves. Consider first the two-dimensional case and the stacked regularization matrix (5.11) implementing the second derivative operator in both directions.

**Theorem 5.3** *Let $L$ be given by (5.11), and let $L_n^{\{2\}} = U_n\,\Sigma_n\,V_n^T$ and $L_m^{\{2\}} = U_m\,\Sigma_m\,V_m^T$ be the SVDs of $L_n^{\{2\}}$ and $L_m^{\{2\}}$, respectively. Then $\|L\,x\|_2 = \|L_D\,x\|_2$ with*

$$L_D = D\,(V_n \otimes V_m)^T, \tag{5.13}$$

*where $D \in \mathbb{R}^{mn\times mn}$ is a nonnegative diagonal matrix satisfying*

$$D^2 = \Sigma_n^2 \otimes I_2 + I_1 \otimes \Sigma_m^2. \tag{5.14}$$

*Furthermore, define the matrix splitting $V_{d_i} = \left(\overline{V}_{d_i}\,, N_{d_i}\right)$ such that $\mathcal{N}(L_{d_i}) = \mathcal{R}(N_{d_i})$ for $i = 1, 2$. Then a basis for $\mathcal{N}(L) = \mathcal{N}(L_D)$ is given by the columns of the matrix*

$$N = N_{d_1} \otimes N_{d_2}. \tag{5.15}$$

PROOF. Inserting the SVDs of $L_n^{\{2\}}$ and $L_m^{\{2\}}$ and using $I_n = V_n V_n^T$ and $I_m = V_m V_m^T$ we obtain

$$\begin{aligned} L &= \begin{pmatrix} L_n^{\{2\}} \otimes I_m \\ I_n \otimes L_m^{\{2\}} \end{pmatrix} = \begin{pmatrix} (U_n \, \Sigma_n \, V_n^T) \otimes (V_m \, V_m^T) \\ (V_n \, V_n^T) \otimes (U_m \, \Sigma_m \, V_m^T) \end{pmatrix} \\ &= \begin{pmatrix} (U_n \otimes V_m) \, (\Sigma_n \otimes I_m) \, (V_n \otimes V_m)^T \\ (V_n \otimes U_m) \, (I_n \otimes \Sigma_m) \, (V_n \otimes V_m)^T \end{pmatrix} \\ &= \begin{pmatrix} U_n \otimes V_m & 0 \\ 0 & V_n \otimes U_m \end{pmatrix} \begin{pmatrix} \Sigma_n \otimes I_m \\ I_n \otimes \Sigma_m \end{pmatrix} (V_n \otimes V_m)^T. \end{aligned}$$

Since the middle matrix consists of two "stacked" diagonal matrices, we can easily determine an orthogonal matrix $Q$ and a diagonal matrix $D$ such that

$$Q^T \begin{pmatrix} \Sigma_n \otimes I_m \\ I_n \otimes \Sigma_m \end{pmatrix} = \begin{pmatrix} D \\ 0 \end{pmatrix}$$

where the diagonal elements of $D$ are nonnegative. Hence

$$L = \begin{pmatrix} U_n \otimes V_m & 0 \\ 0 & V_n \otimes U_m \end{pmatrix} Q \begin{pmatrix} D \\ 0 \end{pmatrix} (V_n \otimes V_m)^T$$

and we obtain $\|L \, x\|_2 = \|L_D \, x\|_2$. The relation

$$D^2 = \begin{pmatrix} D \\ 0 \end{pmatrix}^T \begin{pmatrix} D \\ 0 \end{pmatrix} = D^T D$$

leads immediately to (5.14). Finally, the validity of the expression for the nullspace is observed directly from Lemma 5.1. □

The consequence of this theorem is that we can substitute the structured matrix $L_D$ for $L$ in the Tikhonov problem (2.15), and that we have an orthogonal basis for the nullspace.

In the proof of Theorem 5.3 it is used that both identity matrices can be expressed in terms of the right singular basis of the $L$ matrix that appear in the corresponding Kronecker column. Therefore, the above approach is only applicable if no more than one matrix in each row and each Kronecker column is different from the identity matrix. It is straightforward to generalize this case to more than two dimensions, and for an $N$-dimensional problem, the Kronecker array for which a generalized form of Theorem 5.3 applies, takes the generic form

$$L_N^{\{s\}} = \begin{pmatrix} I_{d_N} \otimes I_{d_{N-1}} \otimes I_{d_{N-2}} \otimes \cdots \otimes I_{d_2} \otimes I_{d_1} \\ Z_{d_N} \otimes I_{d_{N-1}} \otimes I_{d_{N-2}} \otimes \cdots \otimes I_{d_2} \otimes I_{d_1} \\ I_{d_N} \otimes Z_{d_{N-1}} \otimes I_{d_{N-2}} \otimes \cdots \otimes I_{d_2} \otimes I_{d_1} \\ \vdots \\ I_{d_N} \otimes I_{d_{N-1}} \otimes I_{d_{N-2}} \otimes \cdots \otimes I_{d_2} \otimes Z_{d_1} \end{pmatrix}, \tag{5.16}$$

where $d_i$, $i = 1, 2, \ldots, N$ is the size of the $i$th dimension. The first row consists solely of identity matrices of sizes corresponding to the $N$ dimensions, and the latter rows contain one matrix different from the identity $Z_{d_i}$ each, and all in distinct positions. Any row can be left out to give a new stacked Kronecker array, and the $Z$ matrices can implement one-dimensional discrete derivatives, weight matrices, etc. Furthermore, there are no restrictions on the applicable boundary conditions.

If we define the SVDs of the $Z_{d_i}$ matrices as $Z_{d_i} = U_i \Sigma_i V_i^T$, $i = 1, 2, \ldots, N$, then following Theorem 5.3 we can construct

$$L_D^{\{s\}} = D(W_{d_N} \otimes W_{d_{N-1}} \otimes \cdots \otimes W_{d_1})^T, \tag{5.17}$$

where $W_{d_i} = V_{d_i}$ if the column of the Kronecker products corresponding to the $i$th dimension contains a $Z_{d_i}$ matrix, and $W_{d_i} = I_{d_i}$ if it solely contains identity matrices. The diagonal matrix $D$ consists of sums of Kronecker products of the $\Sigma_i$s and identity matrices. Again $\|L_N^{\{s\}} x\|_2 = \|L_D^{\{s\}} x\|_2$, and the nullspace is easily identified. Following Lemma 5.2, it is seen that if one block row consists solely of identity matrices, then the nullspace is empty. Furthermore, the nullspace is given by $\mathcal{N}(L_N^{\{s\}}) = \text{span}\{\, w_{d_N} \otimes w_{d_{N-1}} \otimes \cdots \otimes w_1 \,\}$ where

$$w_i \in \begin{cases} \mathcal{N}(Z_{d_i}) & \text{if } Z_{d_i} \text{ appear in any row of (5.16)} \\ \mathbb{R}^{d_i} & \text{otherwise} \end{cases}.$$

It is important that the pseudoinverse of the regularization matrix is also easily determined. This is given by the following lemma for the case where the regularization matrix is given by (5.16).

**Lemma 5.4** *The Moore-Penrose pseudoinverse of $L_N^{\{s\}}$ of the form* (5.16) *is given as*

$$L_N^{\{s\}\dagger} = (W_{d_N} \otimes W_{d_{N-1}} \otimes \cdots \otimes W_{d_1}) D^\dagger,$$

*where $W_{d_i}$ and $D$ are from* (5.17).

Proof. The lemma is easily proved by checking the four Moore-Penrose conditions, see Definition 2.5. □

A slightly different factorization of a stacked Kronecker product can be obtained by using the GSVD. Inspired by van Loan [91], we formulate another theorem, similar to Theorem 5.3 for the two-dimensional case (5.11).

**Theorem 5.5** *Let $L$ be given by the stacked Kronecker product matrix*

$$L = \begin{pmatrix} B_1 \otimes B_2 \\ C_1 \otimes C_2 \end{pmatrix},$$

*and assume that the two matrices $(B_1^T, C_1^T)^T$ and $(B_2^T, C_2^T)^T$ have full column rank and that they have more rows than columns. Define the two GSVDs*

$$\begin{matrix} B_1 &=& U_n \Sigma_n \Theta_n^{-1} \\ C_1 &=& V_n M_n \Theta_n^{-1} \end{matrix} \quad \text{and} \quad \begin{matrix} B_2 &=& U_m \Sigma_m \Theta_m^{-1} \\ C_2 &=& V_m M_m \Theta_m^{-1} \end{matrix}.$$

*Then* $\|Lx\|_2 = \|L_D\|_2$ *with* $L_D = D(\Theta_n \otimes \Theta_m)^{-1}$ *where* $D \in \mathbb{R}^{mn \times mn}$ *satisfies*

$$D^2 = (\Sigma_n^T \Sigma_n \otimes \Sigma_m^T \Sigma_m) + (M_n^T M_n \otimes M_m^T M_m). \tag{5.18}$$

PROOF. Insert the GSVDs of the matrix pairs $(B_1, C_1)$ and $(B_2, C_2$ into $L$

$$\begin{aligned} L^{\{s\}} &= \begin{pmatrix} B_1 \otimes B_2 \\ C_1 \otimes C_2 \end{pmatrix} = \begin{pmatrix} (U_n \Sigma_n \Theta_n^{-1}) \otimes (U_m \Sigma_m \Theta_n^{-1}) \\ (V_n M_n \Theta_n^{-1}) \otimes (V_m M_m \Theta_m^{-1}) \end{pmatrix} \\ &= \begin{pmatrix} U_n \otimes U_m & \\ & V_n \otimes V_m \end{pmatrix} \begin{pmatrix} \Sigma_n \otimes \Sigma_m \\ M_n \otimes M_m \end{pmatrix} (\Theta_n \otimes \Theta_m)^{-1}. \end{aligned}$$

We can easily determine an orthogonal matrix $Q$ such that

$$Q \begin{pmatrix} \Sigma_n \otimes \Sigma_m \\ M_n \otimes M_m \end{pmatrix} = \begin{pmatrix} D \\ 0 \end{pmatrix}.$$

Hence, by discarding the orthogonal factors, we obtain

$$L_D = D\,(\Theta_n \otimes \Theta_m)^{-1},$$

and the relation $\|Lx\|_2 = \|L_D x\|_2$. Furthermore, we have

$$D^2 = \begin{pmatrix} D \\ 0 \end{pmatrix}^T \begin{pmatrix} D \\ 0 \end{pmatrix} = \begin{pmatrix} \Sigma_n \otimes \Sigma_m \\ M_n \otimes M_m \end{pmatrix}^T \begin{pmatrix} \Sigma_n \otimes \Sigma_m \\ M_n \otimes M_m \end{pmatrix},$$

which leads to (5.18), and concludes the proof. □

Both Theorems 5.3 and 5.5 are based on the two-dimensional case (5.11) and therefore the applications of the two factorizations overlap each other. But the generalizations are different. The GSVD approach does not require any of the elements of the stacked Kronecker products to be identity matrices as long as the stack for each dimension has full column rank. On the other hand, the GSVD approach is only applicable for stacks with two rows. Therefore, in general Theorem 5.5 is applicable for matrices of the form

$$L^{\{s\}} = \begin{pmatrix} B_N \otimes B_{N-1} \otimes \cdots \otimes B_1 \\ C_N \otimes C_{N-1} \otimes \cdots \otimes C_1 \end{pmatrix}, \tag{5.19}$$

where each stack $(B_i^T, C_i^T)^T$ has full column rank. A straightforward generalization of Theorem 5.5 implies that $\|L^{\{s\}}x\|_2 = \|L_D x\|_2$ where

$$L_D = D\left(\Theta_{d_N} \otimes \Theta_{d_{N-1}} \otimes \cdots \otimes \Theta_1\right)^{-1}.$$

To use iterative regularization, we formally need the Moore-Penrose pseudoinverse for solving systems with $L^{\{s\}}$ and $L^{\{s\}T}$. While the $\Theta_{d_i}$s from the GSVDs are nonorthogonal then the Moore-Penrose pseodoinverse is *not* simply given as $\left(\Theta_{d_N} \otimes \Theta_{d_{N-1}} \otimes \cdots \otimes \Theta_1\right) D^\dagger$. We need to project the pseudoinverse onto the range of $L^{\{s\}}$.

**Theorem 5.6** *Let $L^{\{s\}}$ be of the form (5.19) and let $L_D = D\overline{\Theta}^{-1}$, with $\overline{\Theta} = \big(\Theta_{d_N} \otimes \Theta_{d_{N-1}} \otimes \cdots \otimes \Theta_1\big)$, be the orthogonal transformation of the factorization arising from a generalization of Theorem 5.5. Then the Moore-Penrose pseudoinverse of $L_D$ is given by*

$$L_D^\dagger = \mathcal{P}_{\mathcal{R}(L_D^T)}\overline{\Theta}D^\dagger,$$

*where $\mathcal{P}_{\mathcal{R}(L_D^T)}$ denotes the projection matrix onto $\mathcal{R}(L_D^T)$.*

Proof. Again, the pseudoinverse is verified by checking the four Moore-Penrose conditions, see Definition 2.5.

## 5.3 Numerical Examples

If we want to solve the general-form problem $\min\{\|b - Ax\|_2 + \lambda^2\|Lx\|_2\}$, we arrive at a delicate problem of choosing the optimal combination of regularization matrix and boundary conditions. We restrict the possibilities to consider only the two main cases from the previous sections, i.e, different types of approximations to a second derivative operator. In the two-dimensional case this leads to the two formulations described above, i.e, either an approximation to $\|\nabla^2 f\|_2^2$, or an approximation to $\|\partial^2 f/\partial x_1^2\|_2^2 + \|\partial^2 f/\partial x_2^2\|_2^2$. The boundary conditions can be either zero, periodic, reflexive, or none, and furthermore, the coefficient matrix $A$ may also implement any of these boundary conditions. We consider the following questions

- Should the boundary condition imposed on $L$ reflect the boundary condition of the PSF matrix $A$?
- Should the boundary conditions of $L$ preferably be free regardless of the boundary conditions of $A$?
- What is the difference between the stencil and the stacked approach for two-dimensional problems?

First, we focus on the difference in choosing boundary conditions for the PSF matrix $A$, and the regularization matrix $L$. We study a simplified one-dimensional deblurring problem and investigate the impact of choosing various combinations of boundary conditions for the PSF matrix $A$ and the regularization matrix $L$. From a computational point of view it may be favorable to choose the same boundary conditions for the PSF matrix as for the regularization matrix in which case fast FFT or DCT schemes can be used to diagonalize both matrices simultaneously. On the other hand, the background for choosing boundary conditions for $A$ and $L$ is different.

Consider the Tikhonov problem $\min_x\{\|b - Ax\|_2^2 + \lambda^2\|Lx\|_2^2\}$. To minimize the residual term as much as possible, the boundary conditions implemented in $A$ should reflect the true extension of $x$ in the solution domain because the true extension has affected the right-hand side $b$. On the other hand, the boundary conditions in

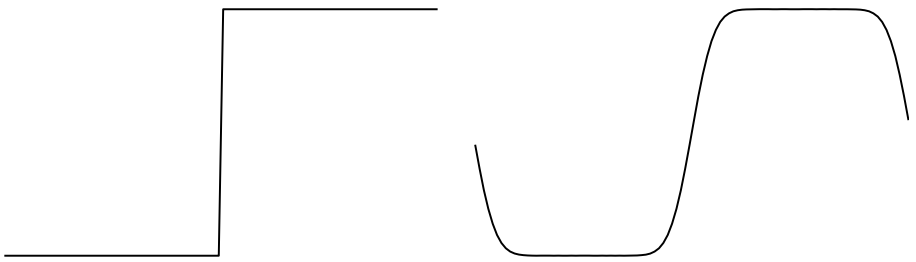


Figure 5.4: Left: one period of true one-dimensional checkerboard image $\bar{x}$. Right: blurred and noisy realization of checkerboard, $\bar{b}^\delta$.

$L$ should be chosen such that the smoothing norm $\|Lx\|_2$ is as small as possible, regardless of how the true extension of $x$ behaves. Consider the following example.

**Example 5.3** *Let the infinite one-dimensional discrete and periodic "checkerboard" image be given by*

$$x_j = \begin{cases} 0 & \text{for} \quad j \in [np\ ;\ np + n/2] \\ 1 & \text{for} \quad j \in [np + n/2 + 1\ ;\ (p+1)n[ \end{cases}, \quad p \in \mathbb{Z} \tag{5.20}$$

*where $n$ is some even number. Moreover, let $A$ implement one-dimensional Gaussian blur such that the elements of $A$ are defined as*

$$a_{i,j} = \frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{1}{2}\left(\frac{(i-j)}{\sigma}\right)^2}, \quad i, j = -\infty, \ldots, \infty, \tag{5.21}$$

*and let the true infinite right-hand side be given as $b = Ax$. Now let $\sigma = 4$ and $n = 100$, and consider only one period of the signal, i.e., $\bar{x}$ is given as (5.20) with $p = 0$, $\overline{A} \in \mathbb{R}^{n \times n}$ is given as (5.21) with $i, j = 1, \ldots, n$, and $\bar{b} \in \mathbb{R}^n$ is the corresponding period of the true right-hand side. Figure 5.4 shows the true image $\bar{x}$ as well as the blurred and noisy right-hand side $\bar{b}^\delta = \bar{b} + e$ where $e \in \mathbb{R}^n$ is white Gaussian noise, scaled such that $\|e\|_2 / \|\bar{b}\|_2 = 10^{-3}$.*

*We now compute Tikhonov solutions $\bar{x}_\lambda = \operatorname{argmin}_{\bar{x}} \{\|\bar{b}^\delta - \overline{A}\bar{x}\|_2 + \lambda^2 \|L\bar{x}\|_2\}$ for a range of $\lambda$s and define the optimal solution as the solution with smallest relative error $\varepsilon_2(\bar{x}_\lambda)$. Furthermore, we first let $L = I$ and implement $\overline{A}$ with zero, periodic, and reflexive boundary conditions. The optimal solutions are seen in Fig. 5.5 (left), and the relative errors are shown in the legend. It is seen that for this problem the best reconstruction is obtained when using periodic boundary conditions for $\overline{A}$. This is to be expected because the sampled right-hand side $\bar{b}^\delta$ is based on the true infinite solution which is in fact periodic.*

*Secondly, we let $\overline{A}$ implement periodic boundary (the optimal choice), and let $L$ be approximations to the second derivative operator (5.4) using zero, periodic, reflexive, or no boundary conditions. The reconstructions are seen in Fig. 5.5 (right), and we*

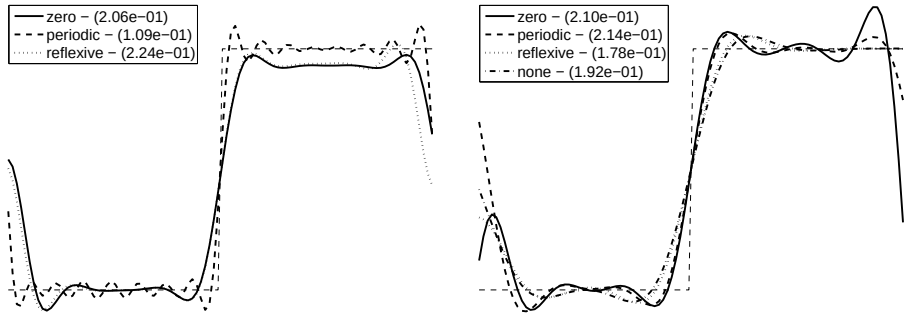



Figure 5.5: Left: reconstructions using $L = I$ and different choices of boundary conditions for $A$. The best obtainable relative errors are denoted in parentesis. Right: $A$ is implemented with periodic boundary conditions, and $L$ is an approximation to the second derivative operator implemented with various boundary conditions.

*note that the optimal reconstruction is achieved when L is implemented with reflexive boundary conditions. We note that L with no boundary conditions (i.e., using* (5.4) *directly) leads to an almost equally good reconstruction. Using the second derivative approximation as regularization term, we try to keep the solutions smooth. As the wanted solution is constant near the boundaries of the domain, but varies a lot from one side of the domain to the other, a reflection of the signal, instead of a periodic extension, is obviously beneficial for the smoothness at the boundaries.*

While the one period "checkerboard" image is not particularly smooth, and the effect of applying the smoothing norm is only significant near the boundaries, it does illustrate in a somewhat brutal way the fundamental difference in choosing boundary conditions for $A$ and $L$. The example clearly illustrates that the boundary conditions of $A$ should reflect the true extension of the signal outside the studied domain, and the boundary conditions of $L$ should be chosen such that the reconstruction near the edges is as good as possible inside the domain. We continue the example a bit further.

**Example 5.4** *Let $\bar{x}$, $\bar{b}$ and $\overline{A}$ be defined as in Example 5.3, and let $\overline{A}$ implement either zero or periodic boundary conditions. Furthermore, let L be an approximation to the second derivative operator* (5.4) *and let it implement reflexive boundary conditions. Now the optimal TGSVD solutions are shown in Fig. 5.6. Note that the behavior of the solutions near the boundaries does not reflect the boundary conditions implemented in $\overline{A}$, but rather the boundary conditions implemented in L. This is obvious because $\overline{A}$ does not describe the solution domain, but the observation domain. That is, $\overline{A}\bar{x}$ must approximate $\bar{b}^\delta$ to make the residual small, and $\bar{b}^\delta$ is definitely not close to zero at the boundaries because it is seriously affected by the true extension*

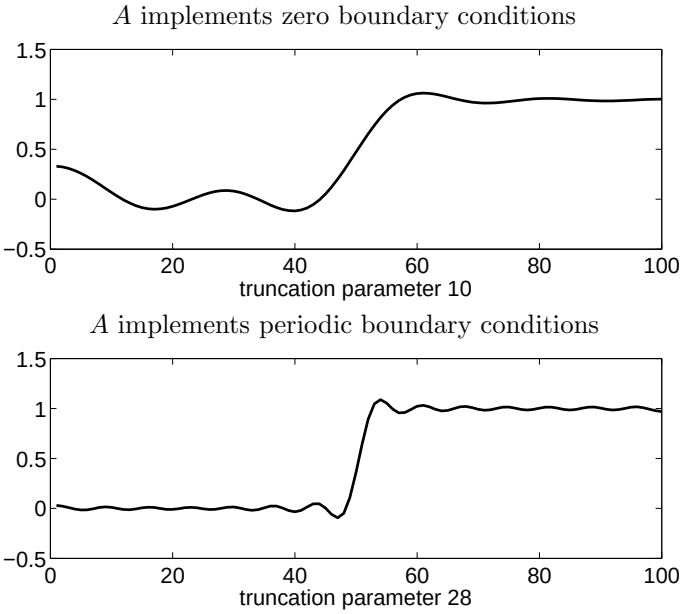



Figure 5.6: Optimal TGSVD solutions using zero or periodic boundary conditions for $A$ and reflexive boundary for the second derivative operator.

*of $x$ (see the right plot in Fig. 5.4). On the other hand, $L$ tries to keep the second derivative small, and because it is implemented with reflexive boundary conditions, this is best achieved if the solution is close to constant at the boundaries, which both regularized solutions are indeed.*

If the true extension of the image is not truly periodic then other boundary conditions for $A$ may be superior. Especially, implementing $A$ with reflexive boundary conditions are studied by several people and found to give pleasing results for more natural two-dimensional images, see, e.g., [69, 74]. Also antireflexive boundary conditions are used [19, 88] and seem to be promising.

While it is beneficial in image deblurring applications to select suitable boundary conditions of the PSF matrix $A$, the use of smoothing norms such as the discrete Laplacian and the Sobolev norm approach, require that the sought solution is expected to be somehow smooth. For natural images, which often contain a lot of contrasts, this is often not the case. Therefore, we use the inverse interpolation problem from Example 4.14 to test the two-dimensional second derivative approximations.

**Example 5.5** *Consider the inverse interpolation test problem from Example 4.14 using the underdeterminded system $Ax = b^\delta$ and the same noise vector.*

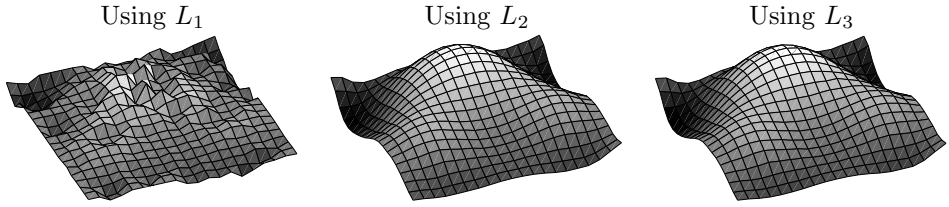



Figure 5.7: Best inverse interpolations using $L_1$, $L_2$ and $L_3$ from (5.22). The relative errors are shows below each surface.

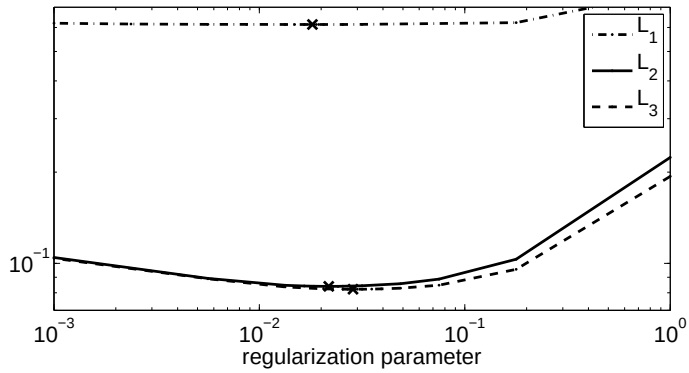



Figure 5.8: Relative errors of the solutions for selected ranges of regularization parameters, and for the three choices of $L$ from (5.22).

*To perform the inverse interpolation we solve the general-form Tikhonov problem* $\min_x\{\|b^\delta - Ax\|_2^2 + \lambda^2\|Lx\|_2^2\}$*, using three choices of* $L$

$$L_1 = I \quad , \quad L_2 = (L_n^{\{2r\}} \otimes I) + (I \otimes L_n^{\{2r\}}) \quad \text{and} \quad L_3 = \begin{pmatrix} L_n^{\{2r\}} \otimes I \\ I \otimes L_n^{\{2r\}} \end{pmatrix}, \tag{5.22}$$

*where* $L_1$ *corresponds to standard-form regularization, and* $L_2$ *and* $L_3$ *implement second derivative smoothing using the two different formulations. Figure 5.7 shows the Tikhonov solutions with minimum relative error, and Fig. 5.8 shows the relative errors as functions of the regularization parameter. We note that the optimal solution to the standard-form problem resembles the LSQR solution from Example 4.14. On the other hand, using general-form regularization with either one of the two formulations of the second derivative operator, we obtain good regularized solutions with similar relative errors.*

## 5.4 Summary

In this section, we introduced regularization matrices for use with general-form regularization for both one and two-dimensional problems. We also studied several generalizations to higher dimensions, and we illustrated that discrete derivative operators in higher dimensions can be difficult to handle. Moreover, we showed how their special structures could be exploited. We also investigated the differences in applying boundary conditions to the coefficient matrix and the regularization matrices. This investigation showed that the background for choosing boundary conditions for the two operators is basically very different. Therefore, to choose similar boundary conditions for $A$ and $L$ such that Tikhonov solutions can be computed directly is might not always be optimal. Finally, we showed that the general-form approach with two-dimensional regularization matrices was able to compute smooth solutions to the inverse interpolation problem.

Chapter 6

# Iterative Regularization of General-Form Problems

In Chapter 4 we studied various Krylov subspace methods applied to $Ax = b$ or $\|b - Ax\|_2$, and saw that sometimes the iterates can be considered as regularized solutions when the problem is ill-posed. In particular CGLS and LSQR have a strong relationship with Tikhonov regularization and TSVD, but also minimum-residual methods such as MINRES and GMRES might produce regularized solutions. However, for some problems standard-form Tikhonov regularization and TSVD do not produce suitable solutions, and it is relevant to compute instead some general-form Tikhonov or TGSVD solution (2.15)–(2.16). We therefore need to consider a seminorm $\|L\,x\|_2$ as the regularization term. But we cannot iteratively compute general-form solutions using a Krylov method, because it cannot see the regularization matrix $L$. We therefore have to reformulate the problem to standard form such that the regularization matrix $L$ is absorbed into the formulation of the residual.

We first summarize some results about the standard-form transformation and how this can been incorporated into CGLS. Next, we illustrate two techniques for extending the concept of smoothing norms to minimum-residual methods, and finally we analyze the methods and illustrate the performance by a number of examples.

## 6.1 Standard-Form Transformation

Consider a general-form Tikhonov problem (2.15) and reformulate this into a Tikhonov problem in standard-form

$$y_{L,\lambda} = \operatorname{argmin}_y \|\bar{b} - \overline{A}y\|_2^2 + \lambda^2 \|y\|_2^2, \tag{6.1}$$

where $y = Lx$. The hope is that a Krylov subspace method applied directly to $\overline{A}y = \bar{b}$ or $\|\bar{b} - \overline{A}y\|_2$ will produce regularized solutions $y^{(k)}$ to the transformed problem (6.1). In turn, the solutions $y^{(k)}$ can then be transformed back to the original solution domain by $x_L^{(k)} = L_\square^\dagger y^{(k)}$ where $L_\square^\dagger$ is some appropriate inverse or pseudoinverse of $L$.

When $L$ is invertible, the standard-form transformation of the Tikhonov problem and the regularized Tikhonov solution $y_{L,\lambda}$ is easy:

$$\overline{A} = AL^{-1} \quad , \quad \bar{b} = b \quad , \quad x_\lambda = L^{-1} y_{L,\lambda}. \tag{6.2}$$

But as described in Section 5.1, for one-dimensional problems we often need regularization matrices $L \in \mathbb{R}^{p\times n}$ that are rectangular with $p < n$ and therefore not invertible, e.g., the discrete approximations to derivative operators (5.3)–(5.4). In these cases the smoothing terms $\|L\,x\|_2$ are seminorms, and the $L$ matrices have nontrivial, but smooth nullspaces. Similar conclusions hold for more complicated regularization matrices, e.g., regularization matrices for two-dimensional problems. To deal with such rectangular matrices, we assume that $L \in \mathbb{R}^{p\times n}$ with $p < n$ has full row rank. Then it was demonstrated in [33] that we should use the transformation

$$\bar{A} = A\,L_A^\dagger, \qquad \bar{b} = b - A\,x_0, \qquad x_{L,\lambda} = L_A^\dagger y_\lambda + x_0,$$

where $L_A^\dagger$ is the $A$-weighted pseudoinverse of $L$, and $x_0$ is the component of the solution in $\mathcal{N}(L)$.

There are several ways to define the matrix $L_A^\dagger$. Eldén [21] used the definition $L_A^\dagger = (I - (A\,(I - L^\dagger L))^\dagger A)\,L^\dagger$. Alternatively, we can use the GSVD from Definition 2.6 on the matrix pair $(A,\,L)$, i.e., $L = V(\,M\,,\,0\,)\,\Theta^{-1}$. If we define the splitting

$$\Theta = (\,\Theta_1\,,\,\Theta_2\,), \quad \Theta_1 \in \mathbb{R}^{n\times p},\, \Theta_2 \in \mathbb{R}^{n\times(n-p)}, \quad \text{and} \quad \mathcal{N}(L) = \mathcal{R}(\Theta_2), \tag{6.3}$$

then $L_A^\dagger$ can also be defined as

$$L_A^\dagger = \Theta_1\,M^{-1}\,V^T = \Theta_1\,(L\,\Theta_1)^{-1} = \Theta_1\,(L\,\Theta_1)^{-1}\,L\,L^\dagger. \tag{6.4}$$

The two definitions of $L_A^\dagger$ are identical, and the matrix

$$E = I - (A\,(I - L^\dagger L))^\dagger A = \Theta_1\,(L\,\Theta_1)^\dagger\,L \tag{6.5}$$

is the oblique projection on $\mathcal{R}(\Theta_1)$ along $\mathcal{R}(\Theta_2) = \mathcal{N}(L)$. (For more on oblique projections, see e.g., [39] and [65, §5.9].) If $L$ is invertible then the weighted pseudoinverse is identical to $L^{-1}$. Moreover, it is identical to the Moore-Penrose pseudoinverse $L^\dagger$ when $p > n$ and $\text{rank}(L) \geq n$. The vector $x_0$ is given by

$$x_0 = \big(A\,(I - L^\dagger L)\big)^\dagger\,b = N\,(A\,N)^\dagger\,b,$$

where the matrix $N$ is any matrix of full column rank such that $\mathcal{R}(N) = \mathcal{N}(L)$.

Hanke and Hansen [33] demonstrated how the CGLS algorithm can be modified in such a way that all operations with $L_A^\dagger$ act as preconditioning. To see this it is shown,

see e.g., [37, §6.1], that if $\overline{\mathcal{P}}_k$ is the solution polynomial after $k$ iterations of CGLS applied to $\overline{A}y = \bar{b}$, then the solution iterate $\bar{x}^{(k)}$ after $k$ steps of the preconditioned CGLS algorithm can be written as

$$\bar{x}^{(k)} = \overline{\mathcal{P}}_k \left( L_A^\dagger L_A^{\dagger T} A^T A \right) L_A^\dagger L_A^{\dagger T} A^T b + x_0, \tag{6.6}$$

and the regularized part of the solution iterates $\bar{x}^{(k)} - x_0$ lie in the Krylov subspace $\mathcal{K}_k(L_A^\dagger L_A^{\dagger T} A^T A,\, L_A^\dagger L_A^{\dagger T} A^T b)$. It is now obvious that $L_A^\dagger L_A^{\dagger T}$ acts like a "preconditioner" for the system, and efficient methods for implementing this kind of preconditioning for CGLS and other methods are described in [33, 36] and [37, §2.3.2]. From the system $\overline{A}^T \overline{A}\, y = \overline{A}^T \bar{b}$ we note that the preconditioning can be identified as a symmetric preconditioning of the normal equations. We will refer to this algorithm as PCGLS.

The efficient implementation of PCGLS from [33, 37] does not directly transfer to an object-oriented implementation where the coefficient matrix $A$ is not necessarily given as a matrix, but might be implemented as a more complicated object. We elaborate on this and device a new implementation of PCGLS for the M$\mathcal{OO}$Re Tools framework in Chapter 8.

Finally, we note that it is always possible to use an orthogonal transformation of $L$ instead of $L$. Let $L = QR$ be a thin QR-factorization such that $R$ is triangular or trapezoidal. Then $\|Lx\|_2 = \|Rx\|_2$, and moreover, $L^\dagger = R^\dagger Q^T$ and $L^\dagger L = R^\dagger R$ such that $L_A^\dagger L_A^{\dagger T} = R_A^\dagger R_A^{\dagger T}$. Therefore the PCGLS iterates are identical because $\mathcal{K}_k(L_A^\dagger L_A^{\dagger T} A^T A, L_A^\dagger L_A^{\dagger T} A^T b) = \mathcal{K}_k(R_A^\dagger R_A^{\dagger T} A^T A, R_A^\dagger R_A^{\dagger T} A^T b)$.

## 6.2 GMRES and Smoothing Norms

We now want to apply a minimum-residual method (e.g., GMRES or MINRES) to the standard-form transformed system $\overline{A}\bar{x} = \bar{b}$, and we note that a primary requirement is that $\overline{A}$ is square. Unfortunately, the commonly used regularization matrices are not square, and therefore $\overline{A} = AL_A^\dagger$ is not in general square. We describe two approaches that address this problem. The first is a simple idea, mainly due to Calvetti et al. [13], and the second is a newly proposed preconditioning technique [40].

### 6.2.1 Augmented-Matrix Approach

To get a square coefficient matrix, Calvetti et al. [13] propose to augment the standard regularization matrices in such a way that the augmented matrix $\widehat{L}$ is square and invertible. Consider the two standard regularization matrices $L_n^{\{1\}}$ and $L_n^{\{2\}}$ (5.3)–(5.4) for one-dimensional problems, and let $w$, $\bar{w} \in \mathbb{R}^n$ be two vectors. Then this

approach results in the following augmented matrices

$$\widehat{L}_n^{\{1\}} = \begin{pmatrix} L_n^{\{1\}} \\ w^T \end{pmatrix} = \begin{pmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \\ w_1 & \cdots & w_{n-1} & w_n \end{pmatrix} \in \mathbb{R}^{n\times n} \tag{6.7}$$

$$\widehat{L}_n^{\{2\}} = \begin{pmatrix} \bar{w}^T \\ L_n^{\{2\}} \\ w^T \end{pmatrix} = \begin{pmatrix} \bar{w}_1 & \bar{w}_2 & \bar{w}_3 & \cdots & \bar{w}_n \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ w_1 & \cdots & w_{n-2} & w_{n-1} & w_n \end{pmatrix} \in \mathbb{R}^{n\times n}. \tag{6.8}$$

We need to choose the additional rows such that the augmented matrices are invertible. In this case (6.2) applies, and therefore we can use $\widehat{L} = \widehat{L}_n^{\{1\}}$ or $\widehat{L} = \widehat{L}_n^{\{2\}}$ and apply GMRES to the system $A\widehat{L}^{-1}y = b$ to produce the solutions $y^{(k)}$. These solutions are then easily transformed back to the solution domain by $x^{(k)} = \widehat{L}^{-1}y^{(k)}$. It is observed that $\widehat{L}^{-1}$ acts like a conventional right preconditioner for GMRES, and we emphasize that this preconditioning is not primarily meant to speed up convergence, but to produce solutions in a more desirable subspace.

Using a polynomial notation, all the back-transformed GMRES iterates $x^{(k)}$ can then be written as

$$x^{(k)} = \mathcal{P}_k\big(\widehat{L}^{-1}A\big)\,\widehat{L}^{-1}b, \tag{6.9}$$

where $\mathcal{P}_k$ is the solution polynomial of GMRES applied to the system $A\,\widehat{L}^{-1}y = b$. The back-transformed solution iterates therefore lie in $\mathcal{K}_k(\widehat{L}^{-1}A, \widehat{L}^{-1}b)$. We stress that we can also apply RRGMRES to the system with the augmented regularization matrix, which results in solution iterates $\hat{x}^{(k)} \in \mathcal{K}_k(\widehat{L}^{-1}A, \widehat{L}^{-1}A\widehat{L}^{-1}b)$.

One must be careful when choosing the extra rows added to the regularization matrix. As illustrated in Section 5.1.1, adding rows is equivalent to adding additional regularization terms. For example, if we replace $\|L_n^{\{1\}}\,x\|_2$ with $\|\widehat{L}_n^{\{1\}}\,x\|_2$ in the Tikhonov problem (2.15), then

$$\|\widehat{L}_n^{\{1\}}\,x\|_2^2 = \|L_n^{\{1\}}\,x\|_2^2 + (w^T x)^2$$

shows that the augmentation of $L_n^{\{1\}}$ is equivalent to adding a second regularization term $\lambda^2\,(w^T x)^2$ to the Tikhonov problem. If the effect of the added row appears close to the boundary, then the added row can be considered as some boundary conditions. The influence of this extra term, and whether it is desirable or not depends heavily on the type of application. For example, we can choose $w = (1, 1, \ldots, 1)^T$ to restrict the sum of the solution elements, and if we want the last solution element to be small then we can use $w = (0, \ldots, 0, 1)^T$, which is equivalent to assuming zero boundary conditions. We can also overcome the choice by incorporating a small weight $\mu$ for

the additional rows such that the contribution to the Tikhonov problem in a practical situation is negligible; i.e., we can use the additional rows $\mu w^T$ and $\mu \bar{w}^T$.

But there are two far more important difficulties with the above mentioned approach.

- Even if the coefficient matrix $A$ is symmetric; the coefficient matrix $A\widehat{L}^{-1}$ is *not* symmetric which excludes the use of the short-recurrence algorithms MINRES and MR-II.
- No orthogonal reduction of the matrix $\widehat{L} = \widehat{Q}\widehat{R}$ can be used because the Krylov subspaces

  $$\mathcal{K}(\widehat{L}^{-1}A, \widehat{L}^{-1}b) = \mathcal{K}(\widehat{R}^{-1}\widehat{Q}^T A, \widehat{R}^{-1}\widehat{Q}^T b) \neq \mathcal{K}(\widehat{R}^{-1}A, \widehat{R}^{-1}b)$$

  are not identical. That is, minimum-residual methods only using $A$ do not solve the least squares problem, and any transformation of $L$ (even orthogonal) will change the problem just like an orthogonal transformation was seen to change the convergence properties of even MINRES and MR-II in Section 4.3.2.

Regarding the symmetry it was proposed in an earlier manuscript by Calvetti et al. [12] to premultiply the system by the transpose of the augmented matrices. If $A$ is symmetric, then the system

$$\widehat{L}^{-T}A\widehat{L}^{-1}y = \widehat{L}^{-T}b, \tag{6.10}$$

can be solved by e.g., MINRES, giving the solution iterates $y^{(k)}$. These solutions are then transformed back by $x^{(k)} = \widehat{L}^{-1}y^{(k)}$. This amounts to a symmetric preconditioning of MINRES, similar to the symmetric preconditioning of the normal equations which is the basis for the PCGLS iterates.

The second issue is harder. Assume that the matrix $L$ has more rows than columns, e.g., a regularization matrix for a 2D-problem of the form (5.11). Now let $L = QR$ be a thin QR factorization such that $R$ is trapezoidal (or triangular if $R$ has full row-rank). Then we can substitute $L$ for $R$ in the general-form Tikhonov formulation because $\|Lx\|_2 = \|Rx\|_2$, but the convergence of some minimum-residual method cannot be expected to be identical. Therefore, the augmented approach is only applicable when using simple augmentation of the standard rectangular regularization matrices.

### 6.2.2 SN Approach

We devise here another technique that works for any rectangular matrix $L$ and which does not introduce any additional constraints or regularization terms in the problem. In addition, our approach naturally preserves symmetry, thus allowing short-recurrence implementations such as MINRES and MR-II to be used if $A$ is symmetric. Also, both GMRES and RRGMRES are still applicable for general systems. Our approach is similar in spirit to the technique for Tikhonov regularization and CG-based

methods for the normal equations. We refer to the new preconditioned algorithms as SN-X, where X = {GMRES, MINRES, MR − II, RRGMRES}, and "SN" is an abbreviation for "smoothing norm."

We consider again a formulation based on the $A$-weighted pseudoinverse of $L$ and start out by writing the solution as the sum of the regularized component in $\mathcal{R}(L_A^\dagger)$ and the unregularized component in $\mathcal{N}(L)$,

$$x = L_A^\dagger y + x_0 = L_A^\dagger y + Nz, \tag{6.11}$$

where again $x_0 = N\,(A\,N)^\dagger\,b$, and $N$ is a matrix with full column rank whose columns span $\mathcal{N}(L)$. These columns need not be orthonormal, although this is preferable for numerical computations. The two vectors $y$ and $z = (A\,N)^\dagger b$ are uniquely determined because $L$ and $N$ both have full rank. Our basic problem $A\,x = b$ can now be formulated as

$$A\left(\,L_A^\dagger\,,\,N\,\right)\begin{pmatrix} y \\ z \end{pmatrix} = b.$$

Premultiplication of this system with $\left(\,L_A^\dagger\,,\,N\,\right)^T$ leads to the $2 \times 2$ block system

$$\begin{pmatrix} L_A^{\dagger T} A\,L_A^\dagger & L_A^{\dagger T} A\,N \\ N^T A\,L_A^\dagger & N^T A\,N \end{pmatrix}\begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} L_A^{\dagger T} b \\ N^T b \end{pmatrix},$$

where we can eliminate $z$ by forming the Schur complement system $S\,y = d$ with $S$ and $d$ given by

$$S = L_A^{\dagger T} A\,L_A^\dagger - L_A^{\dagger T} A\,N(N^T A\,N)^{-1} N^T A\,L_A^\dagger = L_A^{\dagger T} P\,A\,L_A^\dagger, \tag{6.12}$$

$$d = L_A^{\dagger T} b - L_A^{\dagger T} A\,N(N^T A\,N)^{-1} N^T b = L_A^{\dagger T} P\,b, \tag{6.13}$$

where we introduced $P = I - AN(N^T AN)^{-1}N^T$. We will study the system $S\,y = d$ in detail.

**Theorem 6.1** *If $\mathcal{R}(L^T)$ and $\mathcal{R}(AN)$ are complementary subspaces then*

$$P = I - AN(N^T AN)^{-1}N^T \tag{6.14}$$

*is the oblique projector onto $\mathcal{R}(L^T)$ along $\mathcal{R}(AN)$.*

Proof. We first look at the matrix $I - P = AN(N^T AN)^{-1}N^T$. For this to be a projector it must be idempotent which is easily seen because

$$(I - P)^2 = AN(N^T AN)^{-1}N^T AN(N^T AN)^{-1}N^T = (I - P).$$

The projector is not symmetric, and hence it is an oblique projector. The projection is easily seen to be onto $\mathcal{R}(AN)$ with $\mathcal{R}(L^T)$ contained in the null space. The assumption that $\mathcal{R}(L^T)$ and $\mathcal{R}(AN)$ are complementary subspaces is necessary to ensure that both $P$ and $I - P$ are oblique projections; see [39]. When this is fulfilled, $P$ must yield an oblique projection onto $\mathcal{R}(L^T)$ along $\mathcal{R}(AN)$. □

Clearly, when we apply GMRES to the Schur system then there exists a polynomial $\mathcal{P}_k$ such that the solution after $k$ iterations is given by

$$y^{(k)} = \mathcal{P}_k\Big(L_A^{\dagger T} P A\, L_A^{\dagger}\Big)\, L_A^{\dagger T} P\, b.$$

The iterate $y^{(k)}$ is transformed back to the solution domain by means of $x^{(k)} = L_A^{\dagger} y^{(k)} + x_0$, and we therefore obtain the SN-GMRES iterates

$$x^{(k)} = \mathcal{P}_k\Big(L_A^{\dagger} L_A^{\dagger T} P A\Big)\, L_A^{\dagger} L_A^{\dagger T} P\, b + x_0, \tag{6.15}$$

showing that $x^{(k)} - x_0$ lies in the Krylov subspace $\mathcal{K}_k(L_A^{\dagger} L_A^{\dagger T} PA,\, L_A^{\dagger} L_A^{\dagger T} Pb)$. Similar to the case for PCGLS in (6.6) the matrix $L_A^{\dagger} L_A^{\dagger T} P$ appears as a "preconditioner" for the system. The immediate observation is that we perform both right and left preconditioning on the system $Ax = b$ with $L_A^{\dagger T} P$ as the left and $L_A^{\dagger}$ as the right preconditioner. But where PCGLS implements a symmetric preconditioning that can be identified as a right preconditioning of the least squares problem, this non-symmetric preconditioning has no direct connection with the least squares problem.

We emphasize that, similarly with CGLS, the purpose of this preconditioning technique is to hopefully provide a more desirable Krylov subspace for the regularized solution. We note that the iterates of RRGMRES take the same form as for GMRES, but with a different solution polynomial $\widehat{\mathcal{P}}_{k+1}$. Thus RRGMRES can also be used on the Schur complement system.

Although the polynomial expressions for the preconditioned CGLS and GMRES methods in (6.6) and (6.15) are similar in essense, the solutions obtained from the two methods are different because the solution subspaces are different, even when $L$ is invertible.

It is important to note that the two main problems with the augmented-matrix approach from the previous section are both dealt with elegantly with this new approach.

**Theorem 6.2** *If the coefficient matrix $A$ is symmetric, then the transformed coefficient matrix $L_A^{\dagger T} P\, A\, L_A^{\dagger}$ is also symmetric.*

Proof. The symmetry of the transformed system follows directly from the symmetry of $PA = \big(I - AN\,(N^T A\, N)^{-1} N^T\big)\, A = A - AN\,(N^T A\, N)^{-1} (AN)^T$. □

- Theorem 6.2 shows that symmetry is preserved. This symmetry allows us to use MINRES and MR-II on the Schur complement system, resulting in SN-MINRES and SN-MR-II.
- The new approach also allows us to use an orthogonal reduction of the matrix $L$. As for PCGLS, we know that if $L = QR$ is a thin QR factorization and $R$ is triangular or trapezoidal and have full row-rank, then $L_A^{\dagger} L_A^{\dagger T} = R_A^{\dagger} R_A^{\dagger T}$ and thus $\mathcal{K}_k(L_A^{\dagger} L_A^{\dagger T} PA,\, L_A^{\dagger} L_A^{\dagger T} Pb) = \mathcal{K}_k(R_A^{\dagger} R_A^{\dagger T} PA,\, R_A^{\dagger} R_A^{\dagger T} Pb)$.

The following theorem is important for the efficient implementation of the SN-X approach. It shows how we can avoid using the $A$-weighted pseudoinverse when working with the Schur system (6.12)–(6.13).

**Theorem 6.3** *If the requirements in Theorem 6.1 are satisfied, then the Schur system $S\,y = d$ given by (6.12)–(6.13) can be written as*

$$L^{\dagger T} P\, A\, L^{\dagger} x = L^{\dagger T} P\, b \tag{6.16}$$

*with $P$ given by (6.14).*

Proof. From the relation $\big(A(I - L^{\dagger}L)\big)^{\dagger} = (ANN^{\dagger})^{\dagger} = N(AN)^{\dagger}$ it follows that the matrix $E$ in (6.5) can be written as $E = I - N(AN)^{\dagger}A$. Moreover,

$$\begin{aligned} A^T(AN)^{\dagger T}N^T P &= A^T(AN)^{\dagger T}N^T - A^T(AN)^{\dagger T}N^T AN(N^T AN)^{-1}N^T \\ &= A^T(AN)^{\dagger T}N^T - A^T(AN)^{\dagger T}N^T = 0 \end{aligned}$$

and therefore $E^T P = \big(I - A^T(AN)^{\dagger T}N^T\big)P = P$. We also have the relation

$$\begin{aligned} P\,A\,N(AN)^{\dagger}A &= \big(I - AN(N^T AN)^{-1}N^T\big)AN(AN)^{\dagger}A \\ &= AN(AN)^{\dagger}A - AN(AN)^{\dagger}A = 0 \end{aligned}$$

and thus $PAE = PA\big(I - N(AN)^{\dagger}A\big) = PA$. Inserting these relations and $L_A^{\dagger} = EL^{\dagger}$ into the Schur system, we obtain (6.16). □

Theorem 6.3 has an important impact on the numerical implementation of our preconditioning technique, because the weighted pseudoinverse $L_A^{\dagger}$ can be replaced by the ordinary pseudoinverse $L^{\dagger}$ in the computations. The weighted pseudoinverse $L_A^{\dagger}$ is needed only in the back-transformation (6.11).

Turning to the details of the implementation, we need to compute $x_0$ efficiently. This is done via a thin QR factorization of the matrix $AN$ which is always "skinny" for the low-dimensional nullspaces associated with the derivative operators:

1. $AN = Q_0 R_0$ (thin QR factorization)
2. $x_0 \leftarrow N\,R_0^{-1}Q_0^T b$.

We also need an efficient technique for multiplications with $P$ from (6.14), which basically amounts to a number of "skinny" matrix-vector products. Using again the QR factorization of $AN$ we obtain

$$AN\,(N^T AN)^{-1}N^T = Q_0 R_0 (N^T Q_0 R_0)^{-1}N^T = Q_0\,(N^T Q_0)^{-1}N^T$$

and thus the product $P\,x$ is computed as

$$P\,x = x - Q_0\,(N^T Q_0)^{-1}N^T x,$$

where a pre-computed factorization of the small square matrix $N^T Q_0$ should be used. The complete algorithm for performing the multiplication $v = L^{\dagger T}PAL^{\dagger}y$ in the SN-X algorithms now takes the form

1. $v_1 \leftarrow A\,(L^\dagger y)$
2. $v_2 \leftarrow Q_0\,(N^T Q_0)^{-1} N^T v_1$
3. $v \leftarrow L^{\dagger T}(v_1 - v_2)$ .

The cost of working with the Schur complement system is therefore for each iteration: one multiplication with $A$, one with $L^\dagger$, one with $L^{\dagger T}$, and one with the oblique projector $P$. The preconditioning technique is feasible when the computation of $L^\dagger y$ and $L^{\dagger T}(v_1 - v_2)$ can be implemented efficiently, and the nullspace $\mathcal{N}(L)$ has low dimension such that multiplication with $P$ is inexpensive. In Chapter 5 we saw different classes of $L$ matrices for multidimensional problems for which this is achieved, i.e., we have seen already that there are classes of regularization matrices $L$ for which the SN-X algorithms will be feasible.

The remaining work, i.e., possible reorthogonalization and update of the residual norm and the solution, etc., is identical to applying the minimum-residual method to a non-preconditioned system.

## 6.3 Analysis of the Algorithms

The approaches described above are developed by the need for doing general-form regularization connected to the solution of the Tikhonov problem in general form. But is this achieved by GMRES or RRGMRES using the augmented-matrix approach or by the SN-X algorithms? And if not, do we get any satisfactory regularized solutions anyway? We study the Krylov subspaces used by the approaches described in this chapter.

### 6.3.1 PCGLS

In Section 4.2.1, we described CGLS as CG applied to the normal equations. Likewise, the PCGLS algorithm implicitly applies CG-iterations to the transformed system $L_A^{\dagger T} A^T A L_A^\dagger y = L_A^{\dagger T} A^T b$, and therefore constructs a solution $\bar{y}^{(k)}$ in the Krylov subspace $\mathcal{K}_k(L_A^{\dagger T} A^T A L_A^\dagger,\, L_A^{\dagger T} A^T b)$. Lemma 4.3 shows that the CGLS solution iterates can be written in a simple way in terms of the SVD of $A$, or equivalently, the eigenvalues of $A^T A$. We therefore investigate the eigenvalue decomposition of $L_A^{\dagger T} A^T A L_A^\dagger$, equivalently the SVD of $A L_A^\dagger$. Consider the GSVD of the matrix pair $(A,\, L)$ from Definition 2.6

$$A = U \begin{pmatrix} \Sigma & \\ & \widehat{I} \end{pmatrix} \Theta^{-1} \quad , \quad L = V(\, M\,,\, 0\,)\Theta^{-1} \quad , \quad \Theta = (\, \Theta_1\,,\, \Theta_2\,), \tag{6.17}$$

where the splitting of $\Theta$ is defined in (6.3). Using the definition of $L_A^\dagger$ from (6.4), we can now write

$$A L_A^\dagger = U \begin{pmatrix} \Sigma & \\ & \widehat{I} \end{pmatrix} \Theta^{-1} \Theta_1 M^{-1} V^T = U \begin{pmatrix} \Gamma \\ 0 \end{pmatrix} V^T, \tag{6.18}$$

where $\Gamma \in \mathbb{R}^{p\times p}$ is the diagonal matrix that contains the generalized singular values, see Definition 2.6. Therefore, the SVD of $AL_A^\dagger$, and in turn the regularized part of the solution, can be expressed simply in terms of the GSVD of the matrix pair $(A,\, L)$.

We now look at the Krylov subspace $\mathcal{K}_k(L_A^\dagger L_A^{\dagger T} A^T A,\, L_A^\dagger L_A^{\dagger T} A^T b)$ that spans the regularized part of the solution $\bar{x}^{(k)} - x_0$. To analyze this Krylov subspace, we again apply the GSVD, and we get

$$L_A^\dagger L_A^{\dagger T} A^T A = \Theta_1 \begin{pmatrix} \Gamma^2 & 0 \end{pmatrix} \Theta^{-1}, \quad \text{and}$$

$$L_A^\dagger L_A^{\dagger T} A^T b = \Theta_1 \begin{pmatrix} M^{-1}\Gamma & 0 \end{pmatrix} U^T b,$$

Now define the vector $\beta = U^T b$, and express the solution in terms of the generalized singular vectors, i.e., the columns of $\Theta$

$$\Theta^{-1} x \in \operatorname{span}\left\{ \begin{pmatrix} M^{-1}\Gamma & 0 \\ 0 & 0 \end{pmatrix} \beta, \begin{pmatrix} M^{-1}\Gamma^3 & 0 \\ 0 & 0 \end{pmatrix} \beta, \ldots, \begin{pmatrix} M^{-1}\Gamma^{(2k-1)} & 0 \\ 0 & 0 \end{pmatrix} \beta \right\}.$$

Then it is easily seen that the solutions can be expressed by

$$\bar{x}^{(k)} = \Theta_1 \overline{\Phi}_k \Sigma^\dagger U_1^T b + \Theta_2 U_2^T b, \quad \overline{\Phi}_k = \overline{\mathcal{P}}_k\left(\Gamma^2\right)\Gamma^2,$$

where $\overline{\Phi}_k \in \mathbb{R}^{p\times p}$ is a diagonal filter matrix and $\overline{\mathcal{P}}_k$ is the solution polynomial of degree $k-1$ of CGLS applied to $AL_A^\dagger y = b$. We obtain a diagonal filter matrix based on the squared generalized singular values also for the back-transformed solution iterates. The behavior of PCGLS is therefore strongly connected to the GSVD analysis.

Finally, we know from Table 4.1 that applying a Galerkin method like CG to the new system of normal equations, we get a solution that satisfies

$$\min \|b - AL_A^\dagger y\|_2 \quad \text{s.t.} \quad y \in \mathcal{K}_k(L_A^{\dagger T} A^T A L_A^\dagger,\, L_A^{\dagger T} A^T b),$$

i.e., we minimize the correct residual $\|b - AL_A^\dagger y\|_2$ from the standard-form transformed Tikhonov problem as expected.

### 6.3.2 Augmented-Matrix Approach

Now assume that we work with an invertible regularization matrix $\widehat{L} \in \mathbb{R}^{n\times n}$ and let us consider GMRES and the augmented-matrix approach, i.e., we apply GMRES to the system $AL^{-1}y = b$, $x = L^{-1}y$. We use again the GSVD of the matrix pair $(A,\, L)$ to describe the SVD of $AL^{-1}$ and get

$$AL^{-1} = U\Sigma\Theta^{-1}\Theta M^{-1} V^T = U\Gamma V^T.$$

From Lemma 4.4 we know that the SVD of $AL^{-1}$ does not in general describe filter factors for the regularized solution. Furthermore, we still need to transform the

regularized solution back to the solution domain. The back-transformed solution $x^{(k)}$ belongs to $\mathcal{K}_k(L^{-1}A,\, L^{-1}b)$, and we can write the solution iterates as

$$x^{(k)} = \Theta\Phi_k\Sigma^\dagger U^T b, \quad \Phi_k = \mathcal{P}_k\left(M^{-1}V^TU\Sigma\right)M^{-1}V^TU\Sigma, \tag{6.19}$$

where $V^TU$ is in general not diagonal, and $M$ and $\Sigma$ appear on either side of $V^TU$. Therefore, the filter of the augmented-matrix approach is not easily described by the GSVD of the matrix pair $(A,\, L)$. But nevertheless, the approach does minimize the correct residual connected to the standard-form transformed Tikhonov problem

$$y^{(k)} = \operatorname{argmin}_y \|b - A\widehat{L}^{-1}y\|_2 \quad \text{s.t.} \quad y \in \mathcal{K}_k(A\widehat{L}^{-1},\, b).$$

The problem is the solution subspace, as was the case with the difference between CGLS and GMRES.

For a symmetric $A$, it was proposed to use $L^{-T}$ as a left preconditioner to get the system (6.10) with the symmetric coefficient matrix $L^{-T}AL^{-1}$. Applying the GSVD, we get $L^{-T}AL^{-1} = VM^{-1}\Theta^TU\Sigma M^{-1}V^T$, and it follows from the symmetry of $L^{-T}AL^{-1}$ that if we define $\Xi = \Theta^TU$ then also $\Xi\Sigma$ must be symmetric. Looking at the back-transformed solution iterates $x^{(k)}$, it is seen that they belong to $\mathcal{K}_k(L^{-1}L^{-T}A,\, L^{-1}L^{-T}b)$, and in terms of the GSVD we get

$$L^{-1}L^{-T}A = \Theta M^{-2}\Xi\Sigma\Theta^{-1} \tag{6.20}$$

$$L^{-1}L^{-T}b = \Theta M^{-2}\Xi U^T b\,. \tag{6.21}$$

Define again the coefficient vector $\beta = U^Tb$ such that the solution is given in terms of the generalized singular vectors as

$$\Theta^{-1}\hat{x}^{(k)} \in \operatorname{span}\left\{M^{-2}\Xi\beta,\, M^{-2}\Xi\Gamma M^{-1}\Xi\beta,\, \ldots,\, M^{-2}\Xi(\Gamma M^{-1}\Xi)^{k-1}\beta\right\},$$

and the solution iterates can be given as

$$x^{(k)} = \Theta\Phi_k\Sigma^\dagger U^T b, \quad \Phi_k = \mathcal{P}_k\left(M^{-2}\Xi\Sigma\right)M^{-2}\Xi\Sigma. \tag{6.22}$$

Here the filter matrix $\Phi_k$ is obviously not diagonal in general. Furthermore, it is in general not even symmetric because $M^{-2} \neq \Sigma$. Short-recurrence methods like MINRES can still be used because the iteration matrix $L^{-T}AL^{-1}$ is indeed symmetric, but the back-transformed solutions can not be described in a simple way by the GSVD of the matrix pair $(A,\, L)$ – even when $A$ is symmetric.

### 6.3.3 SN-X Approaches

We turn now to the SN-X approaches and use a GMRES-like algorithm on the system $L_A^{\dagger T}PAL_A^\dagger y = L_A^{\dagger T}Pb$, where $A \in \mathbb{R}^{n\times n}$ is a general, possibly nonsymmetric, matrix. The back-transformed solution iterates satisfy $x^{(k)} - x_0 \in \mathcal{K}_k(L_A^\dagger L_A^{\dagger T}PA,\, L_A^\dagger L_A^{\dagger T}Pb)$, and using the GSVD (6.17), we get

$$L_A^\dagger L_A^{\dagger T}PA = \begin{pmatrix}\Theta_1 M^{-2}\Theta_1^T PU_1\Sigma & 0\end{pmatrix}\Theta^{-1}, \quad \text{and}$$

$$L_A^\dagger L_A^{\dagger T} Pb = \begin{pmatrix} \Theta_1 M^{-2} \Theta_1^T P U_1 & 0 \end{pmatrix} U^T b.$$

Now define the matrix $\Xi = \Theta_1^T P U_1 \in \mathbb{R}^{p \times p}$ as well as the coefficient vector $\beta = U^T b$ such that we can express the solution in terms of the generalized singular vectors as

$$\Theta^{-1} \hat{x}^{(k)} \in \operatorname{span} \left\{ \begin{pmatrix} M^{-2}\Xi & 0 \\ 0 & 0 \end{pmatrix} \beta, \begin{pmatrix} M^{-2}\Xi\Gamma M^{-1}\Xi & 0 \\ 0 & 0 \end{pmatrix} \beta, \ldots \right.$$
$$\left. \begin{pmatrix} M^{-2}\Xi(\Gamma M^{-1}\Xi)^{k-1} & 0 \\ 0 & 0 \end{pmatrix} \beta \right\}.$$

The solutions to the back-transformed problem can now be written

$$x^{(k)} = \Theta_1 \Phi_k \Sigma^\dagger U_1^T b + \Theta_2 U_2^T b, \quad \Phi_k = \mathcal{P}_k \left( M^{-2}\Xi\Sigma \right) M^{-2}\Xi\Sigma, \tag{6.23}$$

which is similar to (6.22), except for the appearance of the nullspace component $\Theta_2 U_2^T b$, and that $\Xi$ is defined differently. The filter matrix $\Phi_k$ is seen to be nondiagonal because $\Xi$ is in general nondiagonal. Furthermore, the exponents of $M$ and $\Sigma$ are not identical because the projector $P$ has been substituted with $A^T$. That is, the SN iterates can not be described by a filter of the GSVD components.

As for PCGLS, we take a look at Table 4.1, and note that the solutions obtained by applying a minimum-residual method to the new transformed problem are given by

$$y^{(k)} = \operatorname{argmin}_y \| L_A^{\dagger T} Pb - L_A^{\dagger T} P A L_A^\dagger y \|_2 \quad \text{s.t.} \quad y \in \mathcal{K}_k(L_A^{\dagger T} P A L_A^\dagger, \, L_A^{\dagger T} Pb),$$

which shows that not only the solution subspace, but also the minimization property of this approach is different from PCGLS. Therefore, we do not solve the least squares problem from the standard-form transformed Tikhonov problem.

Assume now that $A \in \mathbb{R}^{n \times n}$ is symmetric. Then we know from Theorem 6.2 that also the new coefficient matrix $L_A^{\dagger T} P A L_A^\dagger$ is symmetric and that we can apply SN-MINRES and SN-MR-II. Again, we insert the GSVD of the matrix pair $(A, L)$, and we get the iteration matrix

$$\begin{aligned} L_A^{\dagger T} P A L_A^\dagger &= V M^{-1} \Theta_1^T P \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \Sigma & \\ & I \end{pmatrix} \Theta^{-1} \Theta_1 M^{-1} V^T \\ &= V M^{-1} \Xi \Sigma M^{-1} V^T, \end{aligned}$$

which shows that $\Xi\Sigma$ must also be symmetric, but *not* necessarily diagonal. Similar to the symmetric augmented-matrix approach (6.22), the filter matrix in (6.23) is in general neither diagonal, nor symmetric, even when the coefficient matrix $A$ is symmetric, and SN-MINRES and SN-MR-II can be used. Therefore, not even in the symmetric case do we get solutions that are easily described in the GSVD domain.

Lemma 4.7 shows that if the coefficient matrix is symmetric, then any method that generate solutions from a Krylov subspace based on this coefficient matrix will generate filtered SVD solutions. Therefore, at first it might seem odd that the SN-MINRES and SN-MR-II methods are *not* spectral filtering methods in the GSVD

basis. But SN-MINRES and SN-MR-II are spectral filtering methods in terms of the SVD of $L_A^{\dagger T} P A L_A^{\dagger}$ and not in terms of the SVD of $L_A^{\dagger} A^T A L_A^{\dagger T}$ as PCGLS. And this is exactly the problem. The SVD of $L_A^{\dagger T} A^T A L_A^{\dagger}$ is determined by the GSVD of the matrix pair $(A,\, L)$ which is not the case for the SVD of $L_A^{\dagger T} P A L_A^{\dagger}$. The nondiagonal $\Xi\Sigma$ and the different powers of $M$ and $\Sigma$ is a result of this difference.

### 6.3.4 Summary

We summarize the systems arising from the different preconditioned algorithms, and provide some general comparisons. The systems are the following

$$\begin{aligned} \text{PCGLS}: &\quad L_A^{\dagger T} A^T A L_A^{\dagger} y = L_A^{\dagger T} A^T b \\ \text{SN}-\text{X}: &\quad L_A^{\dagger T} P A L_A^{\dagger} y = L_A^{\dagger T} P b \\ \text{Aug.}-\text{matrix appr.}: &\quad A\widehat{L}^{-1} y = b, \end{aligned}$$

Note that PCGLS and SN-GMRES look quite similar only changing $A^T$ to $P$, and that the augmented matrix approach stands out a bit. Now for a noise-free right-hand side $b = Ax = \sum_{i=1}^{n} \sigma_i (v_i^T x) u_i$, we consider the first Krylov vectors for the three methods, i.e., the right-hand sides of the above systems.

For PCGLS the right-hand side of the system is $L_A^{\dagger T} A^T b = L^{\dagger T} E^T A^T b$ where $E^T$ is the oblique projector from (6.5). Due to the decaying singular values and a possible numerical nullspace, $A^T b = A^T A x = \sum_{i=1}^{n} \sigma_i^2 (v_i^T x) v_i$ lies in $\mathcal{R}(A^T)$ and will primarily be spanned by the first right singular vectors. Next, $A^T b$ is multiplied by $E^T$, and we take a closer look at the oblique projector $E^T$.

**Theorem 6.4** *Let the GSVD of the matrix pair $(A,\, L)$ be defined as in Definition 2.6 and let $E = \Theta_1 (L\Theta_1)^{\dagger} L$ be the oblique projector from* (6.5)*. Then $E^T = L^T (\Theta_1^T L^T)^{\dagger} \Theta_1^T$ is an oblique projector onto $\mathcal{R}(L^T)$ along $\mathcal{R}(A^T A N)$.*

Proof. First note that $\mathcal{R}(\Theta_1)$ and $\mathcal{R}(\Theta_2)$ are complementary subspaces which implies that $E^T$ is an oblique projector when $E$ is an oblique projector. Then define $W = \Theta^{-T} = (W_1\,,\, W_2)$ where the splitting of $W$ is consistent with the splitting of $\Theta$ and note that $E^T$ projects onto $\mathcal{R}(L^T)$ along $\mathcal{N}(\Theta_1^T) = \mathcal{R}(W_2)$. Finally, it is trivially seen that

$$A^T A N = W \begin{pmatrix} \Sigma^2 & \\ & I \end{pmatrix} \Theta^{-1} \Theta_2 = W_2$$

which concludes the proof. □

Theorem 6.4 shows that $E^T$ projects away from any component of $A^T b$ lying in $\mathcal{R}(A^T A N)$, i.e., all components of $A^T A x$ arising from $x \in \mathcal{N}(L)$ are removed. Furthermore, we project onto $\mathcal{R}(L^T)$ which is the domain of the regularized part of the solution, e.g., $y$ from (6.11).

In case of SN-X, we can write $L_A^{\dagger T} Pb = L^{\dagger T} Pb$ using Theorem 6.3. The original right-hand side $b = Ax = \sum_{i=1}^{n} \sigma_i (v_i^T x) u_i$ is not multiplied by $A^T$, and due to the decaying singular values and a possible numerical nullspace then $b \in \mathcal{R}(A)$ will be primarily spanned by the first *left* singular vectors. Next, $b$ is projected by the projector $P$ which from Theorem 6.1 is known to be the oblique projector onto $\mathcal{R}(L^T)$ along $\mathcal{R}(AN)$, i.e., a projection away from any components of $b = Ax$ arising from $x \in \mathcal{N}(L)$. It is interesting to note, that the projectors $P$ and $E^T$ are in fact very similar. They project onto the same subspace, $\mathcal{R}(L^T)$, but $E^T$ projects along a direction connected with the normal equations and $P$ does not. The vector $E^T A^T b$ from PCGLS has only tiny components in the numerical nullspace of $A$, $\mathcal{N}(A)$, due to numerical round off, whereas the vector $Pb$ from the SN-X algorithms lies in $\mathcal{R}(A)$ which, for a nonsymmetric problem, might include as well significant nullspace components.

Finally, the right-hand side of the augmented approach is simply $b$ which is not projected away from anything because $\mathcal{N}(\widehat{L}) = \{0\}$.

Based on the previous sections and the above comparisons, we note that despite apparent similarities between PCGLS and the SN-X system there are some fundamental differences. The following section provides a number of examples that illustrate how sometimes the SN-X and augmented matrix approach are indeed able to provide suitable regularized solutions, but how sometimes they are not.

## 6.4 Numerical Examples

We know now that just like a standard minimum-residual method only using $A$ may not produce regularized solutions similar to CGLS and Tikhonov solutions, then neither a minimum-residual method using the augmented-matrix approach nor the SN-X algorithms produce solutions that can be described in simple ways as general-form Tikhonov solutions.

Nevertheless, it might in some cases be beneficial to apply the above mentioned approaches. Two examples on this (one symmetric and one nonsymmetric) were developed for the paper [40] (see Appendix C). In the following, we extend the symmetric problem to show how especially SN-MR-II may be a computationally attractive alternative to PCGLS. We also show another nonsymmetric test problem which supports the analysis – i.e., the produced solutions do not need to resemble general-form solutions.

**Example 6.1** *We consider a symmetric two-dimensional test problem. Let the coefficient matrix* $A \in \mathbb{R}^{375000 \times 375000}$ *be a Kronecker product of* $A_1 \in \mathbb{R}^{250 \times 250}$ *and* $A_2 \in \mathbb{R}^{150 \times 150}$ *which are generated from the* deriv2 *test problem from MOORe Tools [49]. The exact solution* $X_{\text{exact}} \in \mathbb{R}^{150 \times 350}$ *is generated by the following* MATLAB *code:*

```
s = linspace(-1,1,250);
t = linspace(-1,1,150);
```

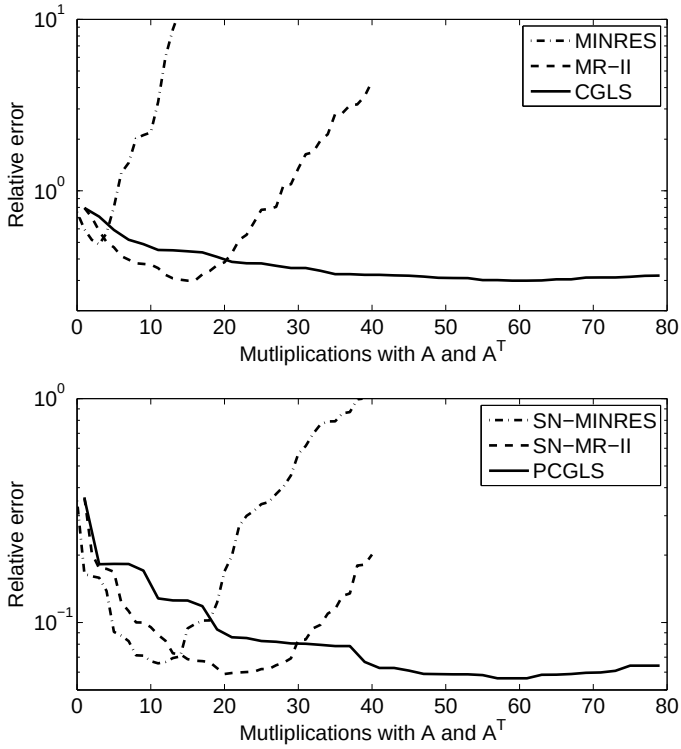



Figure 6.1: Two-dimensional deriv2 test problem. Top: relative errors of MINRES, MR-II and CGLS compared to the number of matrix-vector multiplications with $A$ and $A^T$. Bottom: similar plot of relative errors for SN-MINRES, SN-MR-II and PCGLS.

```
[s,t] = meshgrid(s,t);
X = s + t.^2;
```

*and* $x_{\text{exact}} = \text{vec}\,(X_{\text{exact}})$ *is obtained by stacking the columns of* $X_{\text{exact}}$*. The noisy right-hand side is constructed as* $b^\delta = Ax_{\text{exact}} + e$*, in which* $e$ *is white Gaussian noise scaled such that* $\|e\|_2/\|Ax_{\text{exact}}\|_2 = 10^{-2}$*.*

*We perform 40 iterations of CGLS, MINRES and MR-II as well as PCGLS, SN-MINRES and SN-MR-II. For the preconditioned methods, we use the regularization matrix*

$$L = \begin{pmatrix} L_{250}^{\{1\}} \otimes I \\ I \otimes L_{150}^{\{1\}} \end{pmatrix}, \tag{6.24}$$

*which corresponds to first derivative smoothing in both directions. In Fig. 6.1 we show the relative errors as a function of the number of matrix-vector multiplications*

*with $A$ and $A^T$. All the preconditioned algorithms produce much better solutions than the standard algorithms. Furthermore, both SN-MINRES and SN-MR-II need much fewer matrix-vector multiplications than PCGLS to reach a minimum relative error, and the minimum relative of SN-MR-II is comparable to the minimum relative error of PCGLS.*

The example above illustrates how the SN-X algorithms can have favorable properties, and especially how SN-MR-II can outperform PCGLS. Moreover, the example uses a regularization matrix that is rectangular with more rows than columns such that the augmented-matrix approach cannot be used.

The final example shows how neither the SN-X approach nor the augmented-matrix approach can in general be assumed to provide general-form solutions.

**Example 6.2** *We consider again the* ilaplace *test problem from MOORe Tools [49] with the nonsymmetric coefficient matrix $A \in \mathbb{R}^{n\times n}$, and we use the second implemented solution which is nonzero in the right part of the domain (see also Example 5.1). We use a discretization with $n = 100$ and the blurred and noisy right-hand side is constructed such that $b^\delta = Ax_{\text{exact}} + e$ where $e$ is scaled such that $\|e\|_2/\|Ax\|_2 = 10^{-4}$. We know from Example 5.1 that standard-form regularization is not well suited for this problem, and that general-form regularization using an approximation to the first derivative operator $L_n^{\{1\}}$ in the smoothing norm is far superior. We use CGLS, PCGLS, GMRES, SN-GMRES, and the augmented-matrix approach with GMRES. For the augmented-matrix approach, we use*

$$\widehat{L}_n^{\{1\}} = \begin{pmatrix} 10^{-8} & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix}, \tag{6.25}$$

*i.e., an additional row is added to the top of $L_n^{\{1\}}$ because the implicit zero boundary conditions suit the left part of the wanted solution, and not the right which is constant one.*

*We run each of the algorithms until a residual of about machine precision is achieved. Figure 6.2 shows that CGLS provides a solution which approach zero in the right part of the domain – indeed very similar to the standard-form Tikhonov solution from Example 5.1. PCGLS, on the other hand, is able to reconstruct the wanted solution which approach the constant one in the right part of the domain, i.e., similar to the general-form Tikhonov solution from Example 5.1.*

*For GMRES and SN-GMRES the situation is quite different. Actually, in this case standard GMRES reconstructs the solution better than SN-GMRES, even though none of them behave as neither standard-form regularization, nor general-form regularization. The situation for GMRES using the augmented-matrix approach is even worse, and the initial iterate is the optimal solution. SN-RRGMRES and using RRGMRES in connection with the augmented-matrix show similar behaviors.*

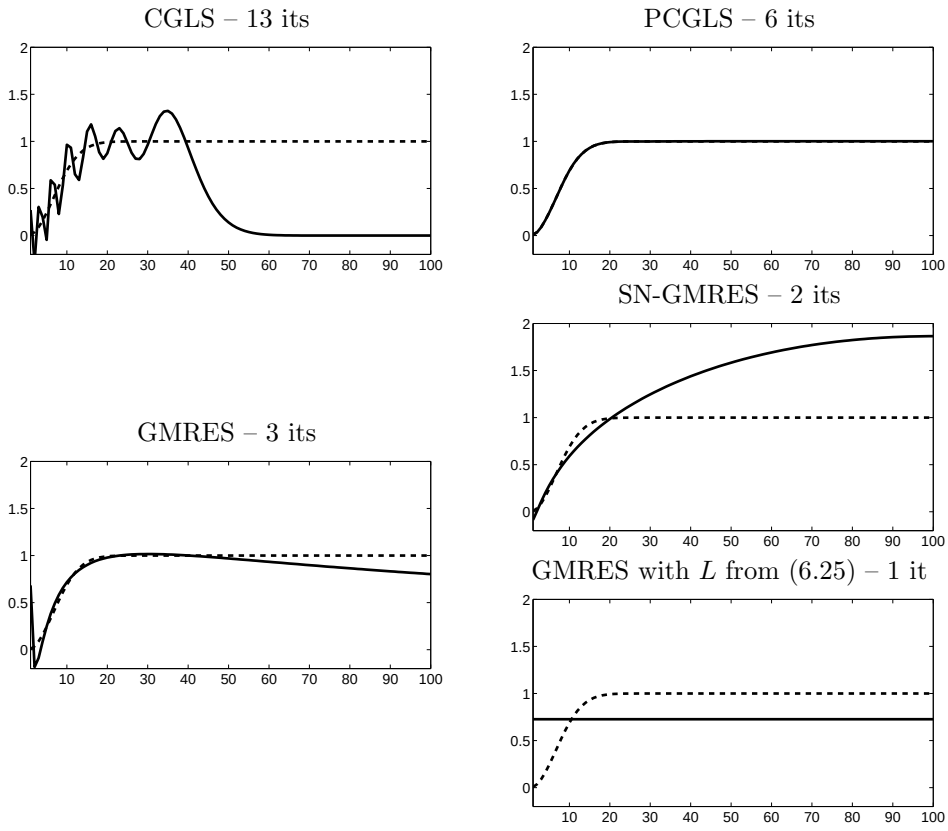



Figure 6.2: The best solution iterates to the ilaplace test problem with second true solution. Methods used are: CGLS, PCGLS, GMRES, SN-GMRES, and GMRES using the augmented-matrix approach. The number of iterates is indicated.

## 6.5 Summary

The concluding remark for this section is just as dubious than for the iterative methods in standard form from Chapter 4. Just like the minimum-residual methods only using $A$ were seen to work for some problems on standard form, then the augmented-matrix approach and the SN-X algorithms may also work for some problems on general form. But there need not be any connection to the general-form Tikhonov problem and none of the methods can be described as filtering methods in the GSVD of the matrix pair $(A, L)$ where $L$ is the regularization matrix.

We did not here look into stopping criteria for the newly developed methods, but it is clear that the issues described in Chapter 4 will also apply here. Therefore, in general we will see no structured relationship between the reduction of the residual and the quality of the reconstructed solution. Moreover, the solution norms need not be monotonically increasing.

Chapter 7

# Discrete L-curves

In Chapter 2 we mentioned several parameter choice methods for locating a suitable regularization parameter that hopefully leads to a good regularized solution. These standard strategies include among others the discrepancy principle and the L-curve.

In Chapter 4 we noted that a stopping rule based on the discrepancy principle is applicable to the CGLS iterates because of the monotonically decreasing residual norms and because the solution components are included in the CGLS iterates in the "correct" order. A similar conclusion was seen to hold for MINRES and MR-II. Moreover, the solution norms are monotonically increasing, i.e., also the L-curve method can be used with CGLS, MINRES, and MR-II. On the other hand, a stopping rule based on the discrepancy principle was seen not to work in general for GMRES and RRGMRES. Moreover, even if GMRES and RRGMRES are able to produce good solutions, the solution norms need not be monotonically increasing. Therefore, the L-curve is not well defined for GMRES and RRGMRES. We will study the so-called condition L-curve [9] in Section 7.4.

In the main part of this chapter, we describe the possible difficulties in finding the corner of a discrete L-curve, and we device a new corner location strategy.

## 7.1 The Discrete L-Curve Criterion

A discrete L-curve, like a continuous one, consists of corresponding values of residual norms and solution (semi-)norms, plotted in log-log scale. For the discrete regularization parameter $k$ and the regularized solutions $x_{L,k}$, we plot corresponding values of $\|b - Ax_{L,k}\|_2$ and $\|Lx_{L,k}\|_2$ where $L$ is some regularization matrix. But contrary

to the continuous case, we here need to find the best corner among the set of discrete points that make up the L-curve. Let in the following $\Pi_k$ denote the $k$th point on a discrete L-curve, i.e., the point $\Pi_k = (\log \|b - Ax_{L,k}\|_2, \log \|Lx_{L,k}\|_2)$. To find the corner in an automated fashion is often not an easy task. When we have only the discrete curve, the second derivatives are not defined and searching for the point of maximum curvature is not directly possible. Furthermore, if the discrete L-curve consists of only a limited number of points then it might be difficult to locate the corner. The distribution of points on the discrete L-curve may also result in small clusters of points. These small clusters might in turn contain small local "corners" that are irrelevant compared to the global corner, but nevertheless disturb a corner finding algorithm.

In the following, we first briefly describe two existing algorithms for locating the corner of discrete L-curves – the l_corner by Hansen and O'Leary [44], and the *triangle method* by Castellanos et al. [16]. The latter algorithm, as well as the new corner location algorithm that will be proposed, are based on a "vector representation" of the discrete L-curve. For future reference, we state the following basic definition.

**Definition 7.1** Consider any two points on the discrete L-curve $\Pi_i$ and $\Pi_j$ for which $i < j$. Then the vector $v_{i,j}$ denotes the vector from $\Pi_i$ to $\Pi_j$. Moreover, the corresponding normalized vector is denoted $\bar{v}_{i,j} = v_{i,j}/\|v_{i,j}\|_2$.

We define also the angle between two vectors, i.e., the angle connected to a triplet of L-curve points $\Pi_j$, $\Pi_k$ and $\Pi_\ell$.

**Definition 7.2** Let $\Pi_j$, $\Pi_k$ and $\Pi_\ell$ be three points on a discrete L-curve satisfying $j < k < \ell$, such that $v_{j,k}$ and $v_{k,\ell}$ by Definition 7.1 are two vectors generated from the discrete L-curve. Then the angle $\theta(j,k,\ell) \in [-\pi;\pi]$ is defined as

$$\theta(j,k,\ell) = \measuredangle(v_{j,k}, v_{k,\ell}).$$

such that an angle $\theta(j,k,\ell) < 0$ corresponds to a point which is a potential candidate for corner point, while $\theta(j,k,\ell) \geq 0$ indicates a point of no interest.

For convenience, we illustrate in Fig. 7.1 how a discrete L-curve is vectorized, and how two different triplets of L-curve points describe a positive and a negative angle, respectively. From this illustration, it is obvious that we seek negative angles $\theta$ as described in Definition 7.2.

The l_corner implementation does not use the vector representation of the L-curve. Instead, Hansen and O'Leary [44] proposed to fit a 2D spline curve to the discrete L-curve and thus create a twice differentiable curve. The point of maximum curvature is then easily computed, and the point on the discrete L-curve with the shortest Euclidean distance to this point is selected as the corner. Unfortunately, the spline curve has a tendency to track small local phenomena which might erroneously result in detection of a wrong corner location. Therefore, the discrete points are first smoothed by local low-degree polynomials. But depending on the small variations,

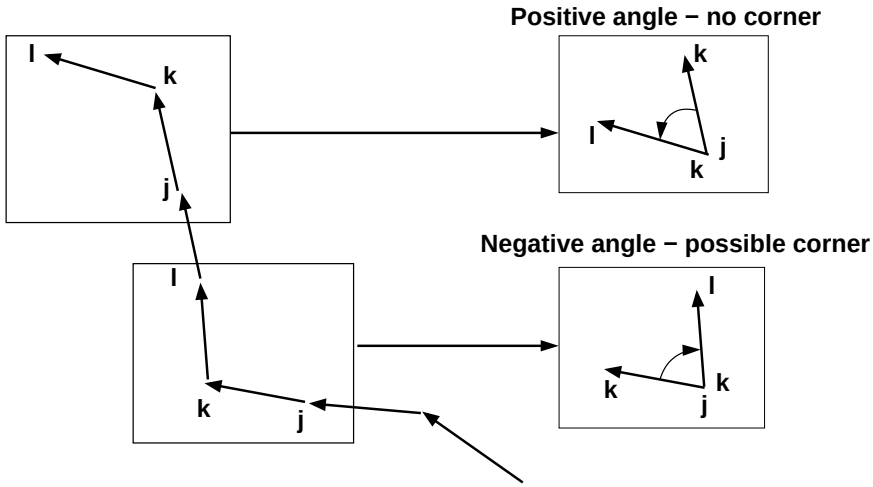



Figure 7.1: Illustration of vectorized discrete L-curve, and how the angles from Definition 7.2 appear as positive and negative, respectively.

different parameters for the low-degree smoothing should be used, i.e., for each new problem we possibly need to adjust the smoothing to correctly identify the corner. This is a serious problem for the robustness of the algorithm. The algorithm is given schematically in Alg. 7.1 and is implemented in l_corner in Regularization Tools [36] and adopted as well in MOORe Tools [49].

**Algorithm 7.1**

***l_corner***
*The overall design of the algorithm l_corner from [36]. The two integers $d$ and $q$ that determine the fit are problem dependent.*

1. *For* $i = q+1, \ldots, p-q-1$
2. *Fit two degree-$d$ polynomials to coordinates of points* $\Pi_{i-q}, \ldots, \Pi_{i+q}$.
3. *Let* $\widehat{\Pi}_i$ = *values of fitting polynomials at* $\Pi_i$.
4. *Fit a 2D spline curve to the new points* $\{\widehat{\Pi}_i\}$.
5. *Compute derivatives of the spline curve at* $\{\widehat{\Pi}_i\}$.
6. *Compute the curvature* $\kappa_i$ *at the points* $\{\widehat{\Pi}_i\}$.
7. *Let* $k = \max_i(\kappa_i)$.

The more recent *triangle method* by Castellanos et al. [16] does use a vector representation of the discrete L-curve. If a discrete L-curve have a total of $p$ points, then the method considers the following triplets

$$(\Pi_j, \Pi_k, \Pi_p)\,, \qquad j = 1, \ldots, p-2, \quad k = j+1, \ldots, p-1, \tag{7.1}$$

i.e., triplets of points consisting of the upper left point of the L-curve, and two additional points. The triangle method considers the angles $\theta(j, k, p)$ from Definition 7.2, defined by all triplets of L-curve points. It selects as the corner the middle point in a triplet for which the angle is minimum. Furthermore, the method checks whether the L-curve has a corner or not. For a more specific description of the algorithm, see [16] and the schematic algorithm given in Alg. 7.2. While the algorithm indeed tries to use a global perspective by not only using consecutive L-curve points, it does have some difficulties as well. First, the complexity is $\frac{1}{2}(p-1)(p-2)$ which is high for large-scale problems where $p$ can be large. The amount of computation can be reduced by working with a subsampled L-curve, but the subsampling must be done carefully by the user and is not part of the algorithm. Secondly, the algorithm depends on the last point of the L-curve, and an unfavorable last point might result in a wrong corner location.

**Algorithm 7.2**

***Triangle algorithm***
*The overall design of the triangle algorithm for finding the corner of a discrete L-curve that consists of a total of p points. The two loops shows a complexity of* $\mathcal{O}(p^2)$.

*Preprocess L-curve by removing zero-norm L-curve points etc.*
*Initialize* $c_{\max} = -2$
*for* $j = 1, \ldots, p-2$
    *for* $k = j+1, \ldots, p-1$
        *Compute vectors:* $v_1 = \Pi_j - \Pi_k$ *and* $v_2 = \Pi_k - \Pi_p$
        *Compute:* $c = \frac{-v_1^T v_2}{\|v_1\|_2 \|v_2\|_2}$
        *Compute:* $a = |v_1 \; v_2|$
        *if* $c > \cos(7\pi/8)$ *and* $c > c_{\max}$ *and* $a < 0$
            *Set:* corner $= k$ *and* $c_{\max} = c$

As seen, both algorithms described above may in some cases have difficulties finding the optimal corner of a discrete L-curve. The following example illustrates some of these issues, as well as the difference between a continuous and a discrete L-curve.

**Example 7.1** *Consider a tiny square inverse problem of size* $n = 12$ *for which the continuous Tikhonov L-curve is shown in Fig. 7.2 (top left). The plot also shows the*

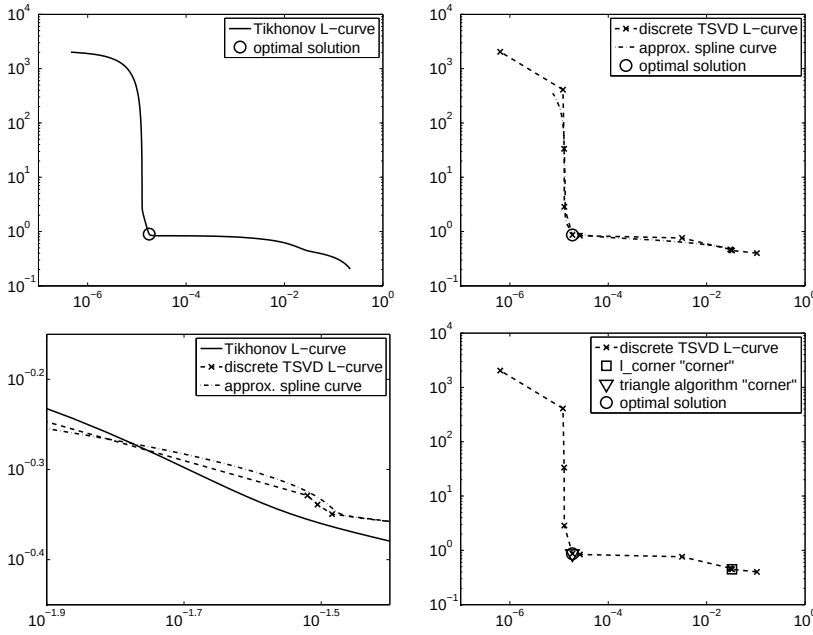



Figure 7.2: Top left: Tikhonov L-curve for small inverse problem of size $n = 12$. The optimal solution is indicated by ∘. Top right: the TSVD L-curve for the same problem. Again a ∘ indicates the optimal solution. The plot also shows the approximate spline curve due to l_corner. Bottom left: zoom of local phenomenon on the horizontal part of the L-curve. The plot shows both the TSVD L-curve, the Tikhonov L-curve, and the default spline approximation produced by l_corner . Bottom right: discrete L-curve with l_corner corner, *triangle* corner and the optimal solution.

*point on the continuous L-curve that corresponds to the regularized solution with the smallest relative error $\varepsilon_2(x_\lambda)$. It is seen that this optimal solution indeed lies near the corner of the L-curve. Figure 7.2 (top right) shows the discrete L-curve for the same problem generated from the TSVD solutions $x_k$ for $k = 1, 2, \ldots, 11$. Again, the solution with smallest relative error is indicated, and obviously, this solution again corresponds to the corner of the discrete L-curve.*

*The default spline approximation of the discrete L-curve from l_corner is determined and also shown in Fig. 7.2 (top right). Figure 7.2 (bottom left) shows a zoom of the discrete and continuous L-curve including the spline approximation. Note that this part of the L-curve belongs to the horizontal part and not the corner. Nevertheless, the spline approximation follows the local variation of the discrete L-curve and erroneously produce a curve with high curvature. Indeed the point of the spline with maximum curvature appears here, and Fig. 7.2 (bottom right) shows the discrete L-*

*curve with the* l_corner *corner and the triangle corner indicated. The triangle method seems to find the correct corner for this problem.*

*We remark that the continuous Tikhonov L-curves do usually not have this drawback, due to the inherent "smoothing" in the expressions for the residual norms and solution norms of the Tikhonov solutions* $x_\lambda$ *compared to the similar expressions for the TSVD solutions* $x_k$*, see* (2.18) *and* (2.19).

## 7.2 New Corner Location Algorithm

A new strategy for locating the corner of a discrete L-curve is proposed. The method is developed with inspiration from Castellanos et al. [16], Rodriguez and Theis [84], and Belge et al. [3]. The method is described here on its own, but the actual implementation is now a part of the adaptive pruning algorithm [41] which is briefly described later.

Following the idea from the triangle method, we consider the discrete L-curve points $\Pi_k$ for $k = 1, 2, \ldots, p$ where $p$ is the number of points on the L-curve. We use the vectors and angles as described in Definitions 7.1–7.2. In principle, it ought to be easy to find the corner of a discrete L-curve directly from Definition 7.2 (see also Fig. 7.1):

1. compute the angle $\theta(k-1, k, k+1)$ for $k = 2, \ldots, p-1$.
2. associate the corner point $\Pi_k$ with the angle closest to $-\pi/2$.

Unfortunately, this simple approach will not work in general as an individual algorithm because discrete L-curves often have several small local corners, occasionally associated with clusters of L-curve points. Furthermore, if the corner consists of a lot of points, e.g., the L-curve for a large-scale problem, then the local angle between any two consecutive vectors need not be close to $-\pi/2$.

Instead a global point of view of the discrete L-curve is needed in order to find the desired corner of the overall curve.

### 7.2.1 The Idea

From a visual perspective, it is often easy to locate the corner of an L-curve, continuous or discrete, unless the L-curve has more than one real corner which indeed may happen. Obviously, a human observer applies an "overall evaluation" of the curvature of the L-curve to identify the corner. To mimic this approach, we must try to extract the overall behavior of the L-curve and not be fooled by local details like the spline approach did in Example 7.1. An example is given in the top left plot of Fig. 7.3 which shows a discrete L-curve for a realization of the test problem `deriv2` from M$\mathcal{OO}$Re Tools. The coefficient matrix is of size $64 \times 64$ and noise is added to the right-hand side such that $\|e\|_2/\|Ax_{\text{exact}}\|_2 = 10^{-2}$. We see from the overall behavior of the curve that the corner is located near the intersection between the two

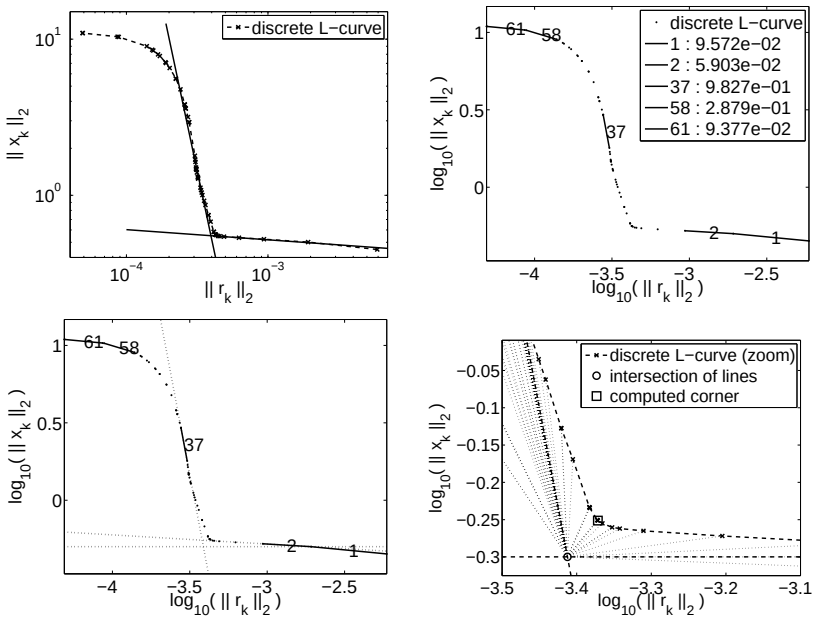



Figure 7.3: Illustration of new corner finding algorithm. Top left: example of L-curve generated by the deriv2 test problem of size 64. The overall behavior of the curve is illustrated by two straight lines. Top right: the five longest vectors are indicated and the legend shows the values of $w_k$ from (7.2). Bottom left: extensions of the vectors are shown to indicate the location of the intersection, and bottom right: zoom of L-curve corner. The intersection is marked by ∘ and all distances are shown by dotted lines. The found corner is indicated by a □.

lines that approximate the first flat part of the curve and the steep part of the curve. If we locate correctly this intersection then we can select the corner of the discrete L-curve as the point with the shortest Euclidean distance to this intersection. The approach is somewhat related to the general approach described in [3] for finding multiple regularization parameters in a Tikhonov framework.

In principle, we could try to fit a number of straight lines to a general discrete L-curve as in Fig. 7.3 top left. But we do not know a priori how many lines are optimal, and the work needed for the optimization problem might be large. We therefore work with vectors as defined in Definition 7.1 to represent the L-curve. The number of vectors needed for finding the corner is not given a priori, and we therefore work with a number of subsampled L-curves where a limited number of L-curve vectors are used.

For some subsampling of the L-curve, the overall algorithm for finding the corner consists of three parts:

1. find vectors that represent the horizontal and vertical parts of L-curve.
2. find the intersection that indicate the location of the corner.
3. locate point with the shortest Euclidean distance to the intersection.

The following section describes the algorithm in more detail.

### 7.2.2 The Method

We illustrate the idea described above more methodically and by an example. Assume that for the discrete L-curve in the top left plot of Fig. 7.3 we choose a subsampling of the L-curve consisting of the $P$ longest vectors.

#### Part I – Horizontal and Vertical Parts

From the chosen vectors, we identify the two that best represent the horizontal and vertical parts of the discrete L-curve. Let $\bar{v}_H = (-1, 0)^T$ be a horizontal vector of unit length, and let $\bar{v}_{k,k+1}$ be a normalized vector as defined in Definition 7.1. Furthermore, we define the slope $\phi_k$ as the angle between $\bar{v}_H$ and $\bar{v}_{k,k+1}$. Now the most horizontal vector is identified by $\ell_h = \text{argmin}_k |\phi_k|$, and the most vertical one by $\ell_v = \text{argmin}_k |\phi_k + \pi/2|$. The size of $\phi_k$ can also be expressed through the determinant of $( \bar{v}_{k,k+1} , \bar{v}_H )$, i.e.,

$$w_k = (\bar{v}_{k,k+1})_1(\bar{v}_H)_2 - (\bar{v}_{k,k+1})_2(\bar{v}_H)_1 = -(\bar{v}_{k,k+1})_2, \tag{7.2}$$

where $(\cdot)_i$ indicate the $i$th component of the vector. We note that $w_k$ carries enough information to determine the most vertical and the most horizontal vector as

$$\ell_h = \text{argmin}_k |w_k| \qquad \text{and} \qquad \ell_v = \text{argmax}_k |w_k| \ .$$

Hence, if we have all the normalized vectors of a subsampled L-curve, then the most horizontal and most vertical vector can be identified directly. We see in the top right plot of Fig. 7.3 how the five longest vectors are selected, as well as the value of $w_k$ for each of these vectors. Obviously, $\ell_h = 2$ and $\ell_v = 37$.

#### Part II – Find Intersection

By now, two vectors are chosen to represent the horizontal and vertical parts of the discrete L-curve, namely $v_{\ell_h,\ell_h+1}$ and $v_{\ell_v,\ell_v+1}$. Now we need a way to extrapolate the vectors so they intersect; hopefully at a point near the corner of the discrete L-curve. The most obvious approach is to extrapolate the vectors directly and find the intersection. But often, over-smoothing of the regularized solution is less critical than under-smoothing, i.e., it is less critical to locate the "corner" slightly off on the horizontal branch, than slightly off on the vertical branch of the L-curve. Therefore, the extension of the $v_{\ell_h,\ell_h+1}$ is done from the right-most point of the vector and parallel to the abscissa, i.e., the parallel line at $\|x_{\ell_h}\|_2$. Figure 7.3 (bottom left) shows

the extrapolation of $v_{\ell_v,\ell_v+1}$, and the two different extensions for $v_{\ell_h,\ell_h+1}$. Obviously, the extrapolation of $v_{\ell_h,\ell_h+1}$ and $v_{\ell_v,\ell_v+1}$ intersect for a higher value of the solution norm than the intersection of the vertical extrapolation and the horizontal line.

**Part III – Locating the Corner**

The corner of the discrete L-curve can now be found as the point on the L-curve with the smallest Euclidean distance to the found intersection. Figure 7.3 (bottom right) shows a zoom of the corner as well as dotted lines connecting all points on the discrete L-curve with the intersection. Obviously, the point with the smallest distance to the intersection (indicated by □) is close to the true corner of the L-curve.

## 7.3 The Adaptive Pruning Algorithm

Rodriguez and Theis [84] published a result regarding a new corner finding algorithm also based on angles between subsequent vectors as defined in Definition 7.2. Furthermore, their algorithm takes care of special shapes of L-curves, e.g., it tries to detect well-posed problems etc.

The adaptive pruning algorithm is a result of combined efforts and is based both on the algorithm by Rodriguez et al., and on the new approach described in the previous section. Furthermore, the adaptive pruning algorithm implements a selective pruning of the L-curve points as well as a strategy for finding the overall "best" corner among a list of corner candidates produced from the two corner finding routines.

In Alg. 7.3, we list a schematic illustration of the adaptive pruning algorithm, and for a more thorough description of the method, we refer to the paper [41], which is given in Appendix C. In the paper, it is verified for a large set of test problems that the adaptive pruning algorithm is capable of locating the corner of the discrete L-curves in a robust way. The algorithm is compared both with the two other corner algorithms (l_corner from Alg. 7.1 and the triangle method from Alg. 7.2), as well with the GCV method, and it is found in general to be more robust than these. Moreover, the complexity of the algorithm is experimentally found to be close to $O(p \log_2 p)$ where $p$ is the total number of points on the discrete L-curve.

Finally, the adaptive pruning algorithm is also used with the residual norms and solution norms of the iterates of CGLS when applied to a large-scale image deblurring problem. Again, the algorithm is capable of finding the corner of the discrete L-curve.

## 7.4 The Condition L-Curve

In the Chapter 4 and 6, we saw that GMRES and RRGMRES have serious problems when used as general regularization methods for nonsymmetric problems. Particularly, we saw that the discrepancy principle in general makes no sense for these methods, because there need not be any connection between the residual norms and the quality of the computed solutions.

**Algorithm 7.3**

***Adaptive Pruning Algorithm***
*The overall design of the adaptive pruning algorithm for locating the corner of a discrete L-curve. Here, $\Pi_k$ denotes a point on the original L-curve, and $\Pi_{k_i}$ denotes a candidate point. Moreover, the L-curve consists of a total of p points.*

*Initialize* $P = \min(5,\, p-1)$
***Stage one:*** *while* $P < 2(p-1)$
  $P = \min(P,\, p-1)$
  *Create a pruned L-curve consisting of the P largest line segments.*
  *For each corner location routine*
    *Locate the corner* $\Pi_k$ *using the pruned L-curve.*
    *Add the corner to the list:* $\mathcal{L} = \mathcal{L} \cup \{\Pi_k\}$.
  $P = 2P$
***Stage two:*** *if* $\#\mathcal{L} = 1$ *then* $k = k_1$*; return.*
*Otherwise for* $i = 1, \ldots, \#\mathcal{L} - 1$
  *Compute the slope* $\phi_i$ *associated with point* $\Pi_{k_i}$ *in* $\mathcal{L}$.
*If* $\max\{\phi_i\} < \pi/4$ *then* $k = \max\{k_i\}$*; return.*
*Otherwise let* $k = \min\{k_i : \phi_i > \pi/4 \wedge \theta(k_{i-1},\, k_i,\, k_{i+1}) < 0\}$.

If we consider the L-curve criterion as a parameter choice method then there is the additional problem that the solution norms $\|\hat{x}^{(k)}\|_2$ and $\|\bar{x}^{(k)}\|_2$ for the GMRES and RRGMRES iterates, respectively, are not necessarily monotonically nondecreasing. Therefore, the standard L-curve is not defined for GMRES and RRGMRES.

Another variant of the L-curve criterion that has been proposed in the literature is the condition L-curve [9]. This criterion deals with the above difficulty with the solution norms, and the basic idea is to consider instead the condition number $\kappa(\overline{H}_k)$ of the Hessenberg matrices $\overline{H}_k$ from the Arnoldi decomposition (4.13) that appear in the construction of the MR-solution (4.15). The condition number $\kappa(\overline{H}_k)$ is nondecreasing and approximates the condition number of the coefficient matrix $A$ [9, Proposition 2.1]. Therefore, when the iteration number $k$ grows, so does $\kappa(\overline{H}_k)$, and eventually the projected problem inherits the ill-conditioning of the original problem $Ax = b^\delta$. The philosophy is that to obtain a good regularized solution, we should balance the reduction of the residual norm and the size of $\kappa(\overline{H}_k)$.

While the condition L-curve at a first glance definitely is a better idea for GMRES and RRGMRES than using the L-curve which is not well defined, the following example shows that the condition number $\kappa(\overline{H}_k)$ (like the residual norm) does not provide much information about the solution quality.

**Example 7.2** *We consider again the* ilaplace *test problem from Example 4.13 and we consider 5 different noise levels* $\|e\|_2/\|b\|_2 = \{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}, 10^{-9}\}$. *For*

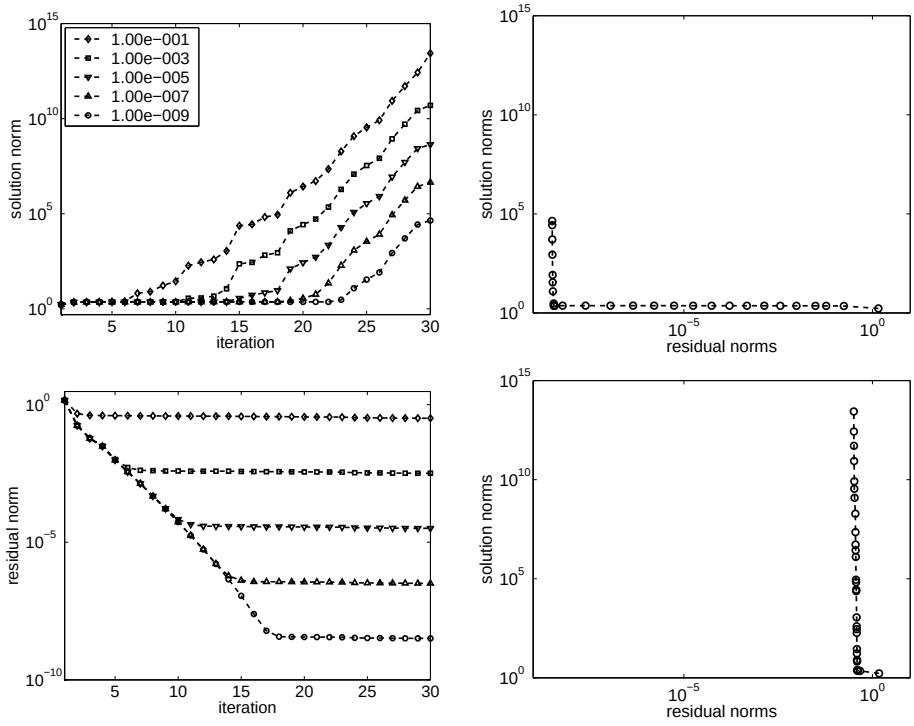



Figure 7.4: Right top: evolution of $\|x^{(k)}\|_2$ and (bottom) $\|b^\delta - Ax^{(k)}\|_2$, $k = 1, 2, \ldots, 30$ for CGLS applied to the ilaplace test problem with different noise levels. Left top: L-curve for CGLS with noise level $\|e\|_2/\|b\|_2 = 10^{-9}$, and left bottom L-curve for CGLS with noise level $\|e\|_2/\|b\|_2 = 10^{-1}$.

*each noise level we compute 30 iterates, $\hat{x}^{(k)}$, of GMRES applied to $Ax = b^\delta$. Moreover, we explicitly construct the Hessenberg matrix $\overline{H}_k$ from* (4.13) *and monitor the condition numbers $\kappa(\overline{H}_k)$ for $k = 1, 2, \ldots, 30$. We also compute the corresponding residual norms $\|b^\delta - A\hat{x}^{(k)}\|_2$ such that we can construct the condition L-curves for the various noise levels.*

*We compute also, for each noise level, 30 iterates, $x^{(k)}$, of LSQR with reorthogonalization of the Krylov vectors. Moreover, we compute the corresponding solution norms $\|x^{(k)}\|_2$ and residual norms $\|b^\delta - Ax^{(k)}\|_2$ such that we can construct the L-curves.*

*Figure 7.4 (left) shows the solution norms and the residual norms for the CGLS iterates, and it is clear that the noise level has a large influence on both residual norms and solution norms. Specifically, we note that a higher noise level leads to an earlier increase of the solution norm, and an earlier stabilization of the residual norm at some constant level. That is, the part of the CGLS iterates that will belong*

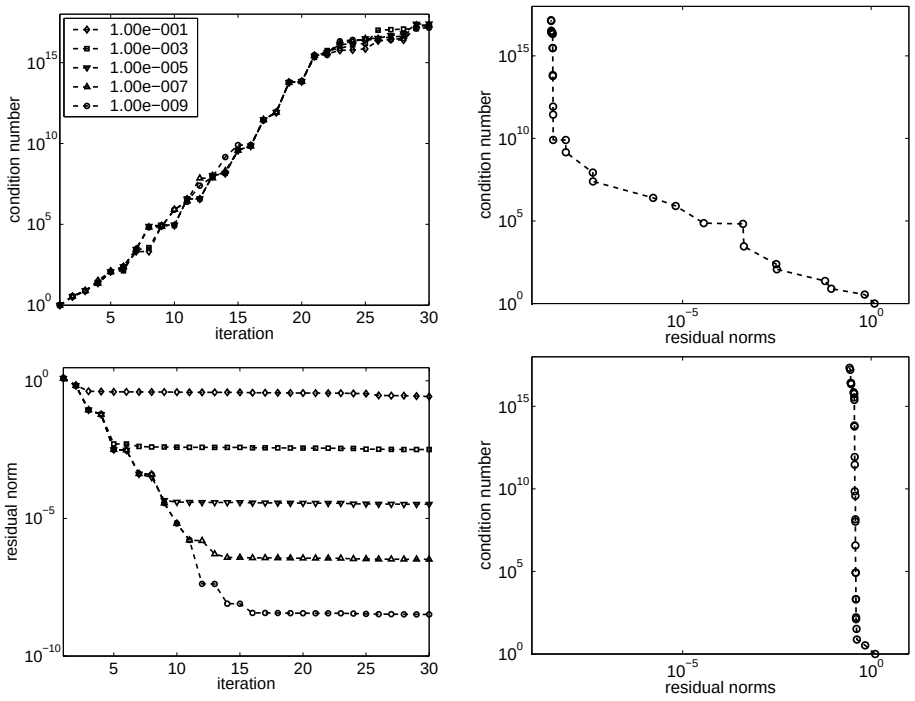



Figure 7.5: Right top: evolution of $\kappa(\overline{H}_k)$ and (bottom) $\|b^\delta - A\hat{x}^{(k)}\|_2$, $k = 1, 2, \ldots, 30$ for GMRES applied to the ilaplace test problem with different noise levels. Left top: condition L-curve for GMRES with noise level $\|e\|_2/\|b\|_2 = 10^{-9}$, and left bottom condition L-curve for GMRES with noise level $\|e\|_2/\|b\|_2 = 10^{-1}$.

*to the horizontal and vertical part of the L-curve, respectively, changes such that the location of the corner changes with the noise level. See Fig. 7.4 (right) for an illustration of the discrete L-curves connected to the smallest and the largest noise level, respectively.*

*For GMRES, see Fig. 7.5 (left), only the residual norms are significantly affected by the different noise levels whereas the evolution of $\kappa(\overline{H}_k)$ is practically unaffected by the different noise levels. This means that the changes in the condition L-curves from one noise level to another are caused by the residual norms. Furthermore, the condition L-curve for the small noise level in Fig. 7.5 (right) has no distinct corner due to the steadily increasing condition number.*

As the above example illustrates, the condition number of the Hessenberg matrices $\overline{H}_k$ from the Arnoldi decomposition do not particularly depend on the noise in the right-hand side. Furthermore, we know from Chapter 4 that the residual norm (and even the number of iterations) need not be correlated with the solution quality. The

"corners" of a condition L-curve are therefore located somewhat randomly, and the use of the condition L-curve is therefore more than questionable.

## 7.5 Summary

This chapter introduced discrete L-curves in general, and presented a new method for locating the corner of such an L-curve. The final implementation of the method is encapsulated in the adaptive pruning algorithm, and the paper [41] describes the algorithm in detail and tests the performance. Moreover, the final implementation of the adaptive pruning algorithm is included as an auxiliary algorithm in M$\mathcal{OO}$Re Tools. We also studied the condition L-curve that has been proposed as a parameter choice method for GMRES, but conclude that there need not be any connection between some corner of a condition L-curve and the quality of the produced solution.

Chapter 8

# MOORe Tools

A not insignificant part of the project has been to maintain and further develop the MATLAB toolbox MOORe Tools by PhD Michael Jacobsen [50], and most implementations for this thesis have been done in this framework. As a first version of the toolbox there are still several problems with some implementations, and writing general code for the toolbox has most often not been an easy task.

This chapter focuses on some of the new implementations in MOORe Tools, and we present three different types: new objects, utility implementations, and algorithms. On the other hand, it is not the intention with this chapter to describe in detail how MOORe Tools is constructed, and neither to describe all the pieces of code that have been written to make everything work. To some extent, Appendices A–B describe how new functionality can be implemented into MOORe Tools as well as some of the problems it gives. Also a list of bug fixes is included.

We do not provide a basic introduction to MOORe Tools, but instead we refer to the PhD thesis [50] which includes a tutorial and a description of the class hierarchy.

## 8.1 New Classes

MOORe Tools is built in an object-oriented fashion. This certainly has a number of advantages because general code can be hidden in high-level classes. In this way, everything that looks like ordinary matrices and vectors on the command prompt can in fact be much more complicated structures. But at implementation level, it certainly adds another level of abstraction, and implementing new basic functionality is not always straightforward. In the following, we describe two new classes.

### 8.1.1 The @StackDerivativeOperator Class

In M$\mathcal{OO}$Re Tools the auxiliary function getL constructs approximations to derivative operators as the ones from (5.3)–(5.4). The @StackDerivativeOperator is a very specialized class that can be used instead of getL for both one-dimensional problems and problems of higher dimensions using the stacked Kronecker approach as covered by Theorem 5.3 and the generalization in (5.16). The actual development of this class is used as a test case and described in more detail in Appendix A.

The class is directly derived form the general superclass @LinearOperator and implements its own matrix-vector multiplication and solve routines in the functions @sub_applytovector.m and sub_solve.m. Furthermore, the functions display.m (which displays information about the object on the command line) and sub_size.m (which returns the size of the object) are implemented specifically for the object such that sensible information will identify the objects on the command prompt.

The most interesting function is sub_solve.m that computes the minimum-norm least squares solutions

```
dia = diag(D.D);
data = getdata(dia);
I = find(data==0); data(I) = 1;
dia = setdata(dia, data);
v = dia.\v; v(I) = 0;
v = D.V*v;
```

where D.D is a diagonal matrix $D$ and D.V is a Kronecker product, both resulting from a factorization as described in Theorem 5.3 or its generalization. A similar operation is performed for the transposed operator. In this way, we implement a special functionality for this particular class, and furthermore, we encapsulate the factorization of the stacked Kronecker products such that a user does not need to worry about the implementation details.

**Example 8.1** *We illustrate the use of the new class with an example. Imagine that we have the following objects*

- *A: Some @LinearOperator working on @Vector2Ds.*
- *b: Right-hand side vector of type @Vector2D.*

*We want to use pcgls (see Section 8.3.1) to implicitly compute general-form solutions $x_L^{(k)}$ using the regularization matrix $L_{mn}^{\{1\}}$, i.e., a stacked Kronecker product that implements the first order approximation to the first derivative operator in both dimensions. In* MATLAB *we do the following*

```
[L,N] = StackDerivativeOperator([m,n], {[], 1 1});
opts = regset('Iter', 10, 'Precond', L, 'Nullspc', N);
xpcg = pcgls(A, b, opts);
```

*which gives the 10th iterate of running cgls on $\|b^\delta - AL_A^\dagger \bar{x}\|_2^2$ transformed back to the solution domain.*

### 8.1.2 The @InverseOperator Class

The @InverseOperator class has been implemented to be able to include inverses and pseudoinverses in a product of operators. First, note that an @OperatorProduct is an object that can contain several other MOORe Tools operators. If for instance $A$ and $L$ are both large and structured matrices that can be implemented as MOORe Tools operators (e.g., @KroneckerProduct2D, @SparseMatrix, @DiagonalOperator, etc.), then the product $AL$ is not necessarily also structured – it might indeed be a general full matrix. For large-scale problems, it might not be possible to construct the product $AL$ and instead we store both $A$ and $L$ in an @OperatorProduct, O=OperatorProduct(A, L);. A multiplication of O with a vector will then result in two object-vector multiplications, first with L, and afterwards with A.

Now, if a series of operators includes an inverse or pseudoinverse

$$v = AL^{\dagger}Pv,$$

it is not possible to directly construct an @OperatorProduct because one of the objects $L^{\dagger}$ is not applied as a product. To get around this problem, the @InverseOperator class has been created such that a multiplication with an object of type @InverseOperator results in a solve with the object stored in the @InverseOperator.

**Example 8.2** *We want to define the object*

$$K = L^{\dagger T}PAL^{\dagger},$$

*using the following MOORe Tools objects*

- *L – @SparseMatrix of size* $p \times n$
- *P – @OperatorProduct with different objects of size* $n \times n$
- *A – @Matrix of size* $n \times n$

*Using the @InverseOperator–class, this can be done by*

```
LI = InverseOperator(L);
K = OperatorProduct({LI', P, A, LI});
```

*without spoiling the structure of the sparse matrices and the already existing @OperatorProduct.*

It is worth to note here that contrary to standard MATLAB, the implementation of the backslash operator for @SparseMatrix and @Matrix in MOORe Tools return the minimum-norm least squares solution for underdetermined systems.

## 8.2 Utility Implementations

### 8.2.1 Picard Plot

Examples of Picard plots were seen in Chapter 2, Fig. 2.2. To show the Picard plot, we need the SVD of the coefficient matrix $A$. This can be computed either if $A$ is small enough or if it has a favorable structure, e.g., if it is a Kronecker product.

In the MOORe Tools framework it is natural to let the function picard.m be a member function of the @SVDOperator class because the Picard plot only makes sense for objects that implement an SVD. The functionality is very similar to the same function from Regularization Tools [36], but the input parameters are adjusted for MOORe Tools and an option for pruning the plot is provided. Pruning the plot can be favorable if the coefficient matrix is large and structured. An example is given below.

**Example 8.3** *We consider a constructed image deblurring problem of an image consisting of* $512 \times 512$ *pixels. The PSF matrix A is represented by a Kronecker product of size* $262144 \times 262144$*, and computing the SVD is only possible due to the favorable structure. We precompute the SVD of A and store the components in an @OperatorSVD so the SVD can be reused. In Fig. 8.1, we show three Picard plots generated from the following code where A is the PSF matrix and b is the noise comtaminated right-hand side*

```
[U,S,V] = svd(A);
OpSVD = OperatorSVD(U,S,V);
picard(OpSVD, b);
picard(OpSVD, b, 400);
picard(OpSVD, b, 400, 5);
```

*The first plot is the default Picard plot where all 262144 components are used. The second plot shows a pruned plot where only 400 components of the 262144 are used, and in the third plot the 400 components are further smoothed by averaging each component with 5 components to each side.*

### 8.2.2 Progress Bars

MOORe Tools is meant for solving large-scale problems, and therefore the iterative algorithms provided by MOORe Tools often run for a long time. The possibility for showing different progress bars has been implemented – one textual, and one graphical.

To implement this kind of functionality, we need to make several changes to MOORe Tools. Firstly, all iterative algorithms are implemented with a common input parameter list through which all options are given in a structure. This structure is generated and used by the utility functions regset and regget, and an extra option – Progress – is therefore implemented. This option can take the values text, bar, off, or

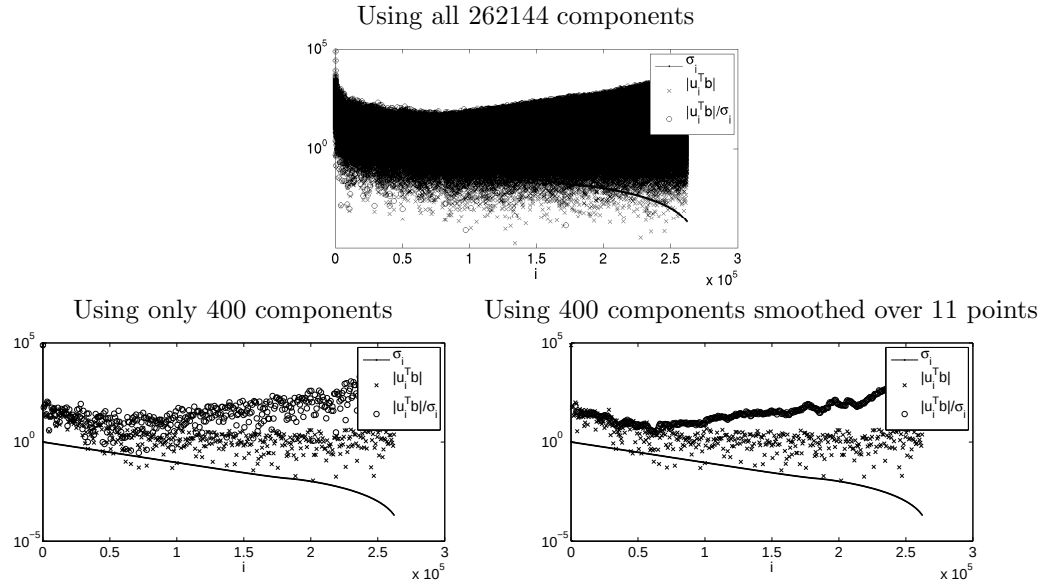



Figure 8.1: Illustration of Picard plots for the constructed image deblurring problem generated by the new MOORe Tools implementation. The top plot performs no pruning of the Picard plot, whereas the bottom plots are pruned and the bottom right plot is also smoothed.

simply []. Moreover, all algorithm must be modified to accept this new option, and the use of the progress bar must be specifically implemented in all algorithms. Finally, the actual progress bar is implemented in the separate utility function progress.m which is placed in the folder Algorithms where auxiliary functions that do not belong to any specific class are placed.

**Example 8.4** *Let A be a @LinearOperator and b be some appropriate vector. Then we can apply 500 LSQR iterations and show a textual progress bar using the code*

```
opts = regset('Iter', 500, 'Progress', 'text');
xlsqr = lsqr_mt(A, b, opts);
```

*which on the* MATLAB *command prompt will show the progress as*

```
Initializing lsqr_mt...
Iteration...
|*************                              | (Max time)
```

*The progress bar shows the maximum number of iterations that can be performed (in this case 500). It might terminate earlier if another stopping criterion is reached, e.g., if the residual norm reaches some desired tolerance.*

## 8.3 Smoothing-Norms for Iterative Methods

Solving a general-form problem with an iterative method can in some cases be done quite efficiently, as noted in Chapter 6. In the following we will study new, efficient implementations of PCGLS and SN-MINRES. Similar approaches for SN-MRII, SN-GMRES, and SN-RRGMRES are straightforward to implement. The implementations are placed under the @LinearOperator folder like all other iterative methods that are based on matrix-vector multiplications only.

### 8.3.1 PCGLS

In case of CGLS, Hanke and Hansen [33] showed that implementing the implicit preconditioning can be done in an efficient way. The approach is used in the PCGLS implementation of Regularization Tools [36], and it is also described in [37, §2.3.2]. Unfortunately, the approach can not be generalized to the object-oriented framework, e.g., because it involves splitting of matrices. This is not possible for the structured objects of MOORe Tools for which only matrix-vector multiplications are guaranteed to work. Therefore, the @WeightedPseudoInverse class had been implemented in MOORe Tools, and the help text to the constructor gives the following example on its use

```
>> [K,b,x] = deriv2(32);
>> [L,W] = getL(32,1);
>> LKinv = WeightedPseudoInverse(L,W,K);
>> x0 = nullcomp(LKinv, b);
>> X = lsqr_mt(K*LKinv, b, regset('Iter', 10));
>> X = LKinv*X + x0;
```

Here the $A$-weighted pseudoinverse is constructed as a separate object LKinv, and the transformed system with the coefficient matrix K*LKinv is solved. Finally, the solution is back-transformed by another multiplication with LKinv. Looking deeper into the implementation of the weighted pseudoinverse we see that this can only be done by performing an extra multiplication with $A$ in each iteration, as well as in the back-transformation.

Before we devise a new implementation strategy which avoids additional multiplications with the coefficient matrix and use the $A$-weighted pseudoinverse in an integrated manner, we summarize in Alg. 8.1 the main CGLS algorithm. Here $d$ denotes the direction in which the solution is updated, and $Ad$ is the update direction of the residuals.

The idea of implicit preconditioning is to apply CGLS to $AL_A^\dagger y = b$, and "on the fly" transform the preconditioned solutions back to the solution domain $x = L_A^\dagger y$. If we substitute $AL_A^\dagger$ for $A$ in Alg. 8.1 then we produce the solutions $y$, and if we at the same time exchange $d$ with $L_A^\dagger d$ in line 5 then we will compute back-transformed solutions. The main idea of the new approach is that the multiplications

**Algorithm 8.1**

***CGLS***
*Schematic illustration of the basic CGLS algorithm.*

1. $r^{(0)} = b$
2. $x^{(0)} = 0$
3. $d = A^T r^{(0)}$
4. ***for*** $j = 1$ *to* $k$
5. $\quad \alpha = \|A^T r^{(j-1)}\|^2 / \|Ad\|^2$
6. $\quad x^{(j)} = x^{(j-1)} + \alpha d$
7. $\quad r^{(j)} = r^{(j-1)} - \alpha A d$
8. $\quad \beta = \|A^T r^{(j)}\|^2 / \|A^T r^{(j-1)}\|^2$
9. $\quad d = A^T r^{(j)} + \beta d$

with $A$ and $A^T$ can be shifted and the results can be reused when the pseudoinverses are applied in an integrated manner. The cost of saving two expensive matrix-vector multiplications for each iteration is that a few additional vectors have to be stored.

First, take a look at the right-hand side of the transformed problem, i.e., the first preconditioned update direction

$$\begin{aligned} d &= L_A^{\dagger T} A^T b \\ &= L^{\dagger T} E^T A^T b = L^{\dagger T}(I - A^T (AN)^{\dagger T} N^T) A^T b \\ &= L^{\dagger T} A^T (I - (AN)^{\dagger T} (AN)^T) b, \end{aligned}$$

where the oblique projector $E^T$ is from (6.5). Note that we can move $A^T$ from the right side of $E^T$ to the left side of the orthogonal projector $(I - (AN)^{\dagger T}(AN)^T)$ because $E^T A^T = A^T (I - (AN)^{\dagger T}(AN)^T)$. Moving the multiplication with $A^T$ from the right side of the projector $E^T$ to the left, we see that now only one multiplication with $A^T$ is needed. The remaining appearances of $A$ are in the combination $AN$ which is often a skinny matrix that can be precomputed. More importantly, $A$ is only applied to the vectors of $N$ from the left which suits the object-oriented framework. Defining $AN = Q_0 R_0$ as the thin QR factorization of $AN$, then

$$d = L^{\dagger T} A^T (I - Q_0 Q_0^T) b \tag{8.1}$$

can be computed very efficiently. When updating the solution, we need to transform this vector back to the solution domain by multiplication with $L_A^\dagger$ such that

$$q = L_A^\dagger d = E L^\dagger d = (I - N(AN)^\dagger A) L^\dagger d = (I - N R_0^\dagger Q_0^T A) L^\dagger d.$$

Now let us carry out the calculation of $q$ in several steps and store the intermediate results such that we can use the partial results.

1. $q_2 = L^\dagger d$
2. $q_3 = Aq_2$
3. $q_4 = Q^T q_3$
4. $q = q_2 - NR^\dagger q_4$,

For calculating the step length $\alpha$ and the update direction of the residual of the least squares problem, we need to form the product $d_r = AL_A^\dagger d$. We get

$$d_r = AL_A^\dagger d = Aq = A(q_2 - NR^\dagger q_4) = q_3 - Q_0 q_4, \tag{8.2}$$

which can be computed very efficiently because the multiplication with $A$ has already been performed above. Finally, we need to update the update direction $d_{\text{new}} = L_A^{\dagger T} A^T r^{(k)} + \beta d$ to get the new solution update direction $L_A^\dagger d_{\text{new}}$ as well as $AL_A^\dagger d_{\text{new}}$. The multiplication with $L_A^{\dagger T} A^T$ can be done efficiently as shown in (8.1)

$$q_1 = L^{\dagger T} A^T (I - Q_0 Q_0^T) r^{(k)}.$$

We now denote the old solution update direction $q_{\text{old}}$ and multiply $q_1$ with $L_A^\dagger$ by exchanging $q_1$ for $d$ in the stepwise procedure shown above. We get

$$L_A^\dagger d_{\text{new}} = L_A^\dagger q_1 + \beta L_A^\dagger d = q + \beta q_{\text{old}},$$

where $q$ is the result of the stepwise procedure for $q_1$. Furthermore, $AL_A^\dagger d_{\text{new}}$ can also be updated by

$$AL_A^\dagger d_{\text{new}} = Aq + \beta d_r = q_3 - Q_0 q_4 + \beta d_r,$$

where step four from the stepwise procedure and (8.2) have been used for the product $Aq$. In this way, all multiplications with $A$ and $A^T$ are performed as a part of applying the $A$-weighted pseudoinverse, and the results can be reused to form the correct PCGLS iterates.

Alg. 8.2 shows a schematic implementation of the modified PCGLS algorithm. The detail level is quite high, and the schematic algorithm is meant to illustrate that in the main loop, there is only one multiplication with $A$ and $A^T$, respectively. Furthermore, the multiplications with $A$ and $A^T$ do not appear in the beginning of the iteration, but they are encapsulated in the formulation of the preconditioned update vectors.

The exact same overall arrangement of the computations can be used also to compute standard CGLS iterates as well as CGLS iterates using a more traditionally preconditioned system with an invertible preconditioner – i.e., solving the least squares problem

$$\min \|AC^{-1}y - b\|_2 \quad , \quad x = C^{-1}y,$$

where $C \in \mathbb{R}^{n \times n}$ is invertible. We note that Alg. 8.2 is shown for the case where $\mathcal{N}(L)$ is nonempty. Therefore, the algorithm simplifies somewhat when using the invertible $C$ instead of the rank-deficient regularization matrices.

**Algorithm 8.2**

***PCGLS***
*Schematic illustration of the MOORe Tools implementation of the PCGLS algorithm.*

1. $AN = Q_0 R_0$ *(thin QR factorization)*
2. $r^{(0)} = b - Q_0 Q_0^T b$
3. $x^{(0)} = N R_0^\dagger Q_0^T b$
4. $v = 0; \quad \beta = 0$
6. $q_1 = L^{\dagger T} A^T r^{(0)}$
7. $q_2 = L^\dagger q_1; \quad q_3 = A q_2 \quad q_4 = Q_0^T q_3 \quad q = q_2 - N R^\dagger Q_0^T q_3$
8. $\gamma = r^{(0)T} A\, q_2$
9. ***for*** $j = 1$ *to* $k$
10. $\quad v = q_3 - Q_0 Q_0^T q_3 + \beta v$
11. $\quad \alpha = \gamma / (v^T v)$
12. $\quad x^{(j)} = x^{(j-1)} + \alpha q$
13. $\quad r^{(j)} = r^{(j-1)} - \alpha v$
14. $\quad s = A^T (r^{(j)} - Q_0 Q_0^T r^{(j)})$
15. $\quad q_1 = L^{\dagger T} s$
16. $\quad q_2 = L^\dagger q_1; \quad q_3 = A q_2; \quad q_4 = Q_0^T q_3;$
17. $\quad \beta = (s^T q_2)/\gamma$
18. $\quad q = q + \beta q_2 - \beta N R^\dagger Q_0^T q_3$

The algorithm pcgls is implemented in MOORe Tools, and it belongs to the base class @LinearOperator like all other basic iterative methods. In this way, any object can be used directly with this new implementation, because all objects are inherited from the @LinearOperator class. The regularization matrix and the operator that contains the basis for the nullspace are given through the options structure. Due to its formulation as a special preconditioner, the regularization matrix is given in the field Precond. Moreover, a new field Nullspc is created in the options structure controlled by regset and regget.

#### 8.3.1.1 Test of Implementation

The new implementation of PCGLS is preferred to the approach using the @WeightedPseudoInverse operator. Both because of its easier use, and above all because it avoids two matrix-vector multiplications with the coefficient matrix in each iteration. The use and speedup is illustrated by the following example.

**Example 8.5** *We use the test problem deriv2 from MOORe Tools for generating problems with coefficient matrices* $A \in \mathbb{R}^{n \times n}$ *of varying sizes,* $n \in [10, 3000]$. *As the*

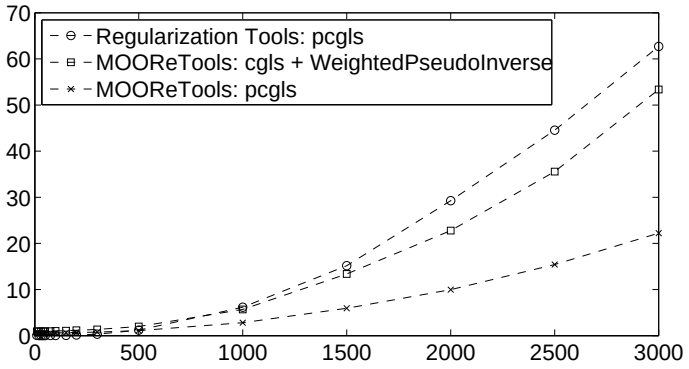



Figure 8.2: Speed test of various PCGLS implementations.

*regularization matrix, we use the approximation of the first derivative operator* $L_n^{\{1\}}$ *from* (5.3)*. We use* pcgls *from* Regularization Tools*, the original* cgls *from MOOReTools in combination with the* @WeightedPseudoInverse*, and finally the new* pcgls *implementation for MOOReTools. We use* MATLAB *code like the following to generate the problem and compute solutions:*

```
>> [A, b, x] = deriv2(n);
>> [L, N] = getL(n, 1);
>> x1 = pcgls(getmatrix(A), getmatrix(L), getmatrix(N), ...
              getvector(b), 30, 1);
>>
>> LA = WeightedPseudoInverse(L,N,A);
>> x2 = cgls(A*LA, b, regset('Iter', 30));
>> x2 = LA*x2 + nullcomp(LA, b);
>>
>> x3 = pcgls(A, b, regset('Iter', 30, 'Precond', L, 'Nullspc', N));
```

*Note how we extract the matrix and vector representations to use* pcgls *from* Regularization Tools*, and how the option structure is used for the new* pcgls *from MOOReTools.*

*Figure 8.2 reports the time spend by each of the different implementations. The shown measurements are mean values for five repetitions of performing 30 iterations with reorthogonalization of the residual directions. It is obvious that the two MOOReTools implementations are both faster than the implementation from* Regularization Tools*, and furthermore, that it indeed pays off to avoid the extra matrix-vector multiplications. We note that for smaller problems, the overhead in using the object-oriented implementations is relatively larger.*

### 8.3.2 SN-X Algorithms

The implementation of the SN approach is a bit different from the PCGLS approach in that $A^T$ is never used, and that $P$ from (6.14) is used instead. First, remember that using the SN-X algorithms, we apply a GMRES-style algorithm to the system

$$L_A^{\dagger T} P A L_A^{\dagger} y = L_A^{\dagger T} P b,$$

and we want to return solutions in the solution domain $x = L_A^{\dagger} y + x_0$ where $x_0$ is the component in $\mathcal{N}(L)$. Basically, there are two different implementation strategies: an object-oriented approach based on representing the transformed system with objects and using the original implementations of GMRES, MINRES, etc., and one where the preconditioning is performed implicitly in specialized implementations.

The object-oriented approach is similar to the approach described for PCGLS where the $A$-weighted pseudoinverse of $L$ is used through a @WeightedPseudoInverse object, and the back-transformation is performed explicitly. The advantage of implementing the SN-X approaches this way is that all four algorithms SN-GMRES, SN-MINRES, SN-RRGMRES, and SN-MRII can be used in a common framework. An implementation of this type is shown in Fig. 8.3, and it has been widely used during the project. As seen, extensive use of objects can be used to form the new coefficient matrix $K = L_A^{\dagger T} P A L_A^{\dagger}$ without ever computing $K$ explicitly. That is, $K$ is formed as one big @OperatorProduct containing another @OperatorProduct, @InverseOperators, an @IdentityOperator, and of course the original coefficient matrix $A$ with whatever structure this might have.

In a specialized implementation, however, we can take advantage of a more efficient use of the matrix-vector multiplications, just like for implementing PCGLS. In principle, implementing the smoothing norm preconditioner is very similar for the different GMRES-style algorithms, and in the following we illustrate the implementation of SN-MINRES.

As for PCGLS, we start by looking at the right-hand side for the transformed problem. Here $A^T$ has been exchanged with the oblique projector $P$, and we get the first Krylov vector

$$v_1 = L_A^{\dagger T} P b = L^{\dagger T} P b = L^{\dagger T}(I - AN(N^T AN)^{-1} N^T) b.$$

We can directly use the thin QR factorization $AN = Q_0 R_0$ to get the computationally simple expression, not involving any large matrix-vector multiplications

$$v_1 = L^{\dagger T}(b - Q_0(N^T Q_0)^{-1}(N^T b)).$$

We do not consider normalization and orthogonalization of the Krylov vectors, but look at each subsequent Lanczos iteration that basically applies the transformed coefficient matrix to the previous Krylov vector. Again, we split up the product $L^{\dagger T} P A L^{\dagger} v_1$ as

1. $q_1 = L^{\dagger} v_1$
2. $q_2 = A q_1$
3. $q_3 = L^{\dagger T} P q = L^{\dagger T} q - Q_0(N^T Q_0)^{-1}(N^T b)$.

```
function [X,extra,restart] = snx(A, b, L, N, method, options, restart)

if nargin<7, restart = []; end
LKinv = WeightedPseudoInverse(L, N, A);
x0 = nullcomp(LKinv, b);
[Q,R] = qr(A*N,0);
NQ = N'*Q;

F = OperatorProduct({Q,InverseOperator(NQ),N'});
P = IdentityOperator(length(b)) - F;
K = OperatorProduct({InverseOperator(L'), P, A, InverseOperator(L)});
y = LKinv'*P*b;

if nargin>6,
   [X,extra,restart] = feval(method,K,y,options,restart);
else
   [X,extra] = feval(method,K,y,options);
   restart = [];
end

for i=1:size(X,2),
    X(:,i) = LKinv*X(:,i) + x0;
end
```

Figure 8.3: MATLAB implementation of SN-X algorithms in a common object-oriented framework.

The Lanczos scaling parameters and the orthogonalization of the Krylov vectors of the transformed system can now be calculated from $q_3$ and the previous vectors. Furthermore, it is also possible to construct vectors that are transformed back to the solution domain without extra matrix vector multiplications. That is, we want to construct the solution update direction $w = L_A^\dagger v_1$, and we get

$$w = L_A^\dagger v_1 = EL^\dagger v_1 = (I - N(AN)^\dagger A)L^\dagger v_1 = q1 - N(R^{-1}(Q^T q_2)),$$

where we reuse the previously computed matrix-vector product as a part of $L_A^\dagger$ in the back-transformation.

The function snminres_mt is implemented in the @LinearOperator folder. The implementation is based on the stepwise procedure described above, and is similar in essence to the PCGLS algorithm in Alg. 8.2.

## 8.4 Summary

In this chapter, we illustrated some of the new MOORe Tools implementations – both new objects, new iterative methods, and new auxiliary methods. We also illustrated by several examples how some basic MOORe Tools constructs look in practise, and how the newly implemented pcgls can be used to compute general-form solutions efficiently.

Chapter 9

# Discussion and Perspectives

This final chapter of the thesis summarizes the results that have been obtained and thereby concludes the work. The project has revolved around several different, but connected, topics. Therefore, the conclusion is likewise divided into several parts. A final section describes some perspectives, and discusses possibilities for future work.

## 9.1 General Conclusion

We studied several aspects of computing regularized solutions to large-scale inverse problems, formulated as linear systems $Ax = b$. We described the matrix structures that arise naturally for image deblurring problems, and showed how large-scale problems of this type can be handled efficiently. In this connection, we studied the relationship between the behavior of the point spread function and the corresponding coefficient matrix – especially regarding boundary conditions and symmetries. We also analyzed various regularization matrices for multidimensional problems; again with a focus on efficient implementation strategies.

The largest part of the work revolved around iterative Krylov subspace methods and their regularization properties. For this purpose, we performed in-depth studies of especially MINRES and GMRES, and generated several illustrative numerical examples. The work has resulted in new important insight into the regularization properties of this class of algorithms. We developed also an extension to existing minimum-residual methods which generalizes the concept of smoothing norms.

Last, but not least, a lot of work has been done to maintain, debug, and further develop the MATLAB toolbox M$\mathcal{OO}$Re Tools. Indeed, several of the developed ideas

have been implemented in the object-oriented framework, and most of the examples and illustrations have been developed using M$\mathcal{OO}$Re Tools.

## 9.2 Iterative Regularization

Regarding the Krylov subspace methods, we provided a general analysis of the basic properties of orthogonal-residual methods and minimum-residual methods. With this basic framework at hand, we studied the methods and found that some are theoretically better suited for application to ill-posed problems than others. Particularly, we performed a study of the underlying Krylov subspaces and analyzed how these can be expressed in the SVD basis – the natural basis for standard-form regularization. The study supported the well-known result that the least squares methods CGLS and LSQR are able to provide filtered SVD solutions. Moreover, the difficulties that arise when applying minimum-residual methods directly to systems of the form $Ax = b$ show up directly in the study of the Krylov subspaces.

We note that whether $A$ is symmetric, normal, or a general matrix has a large influence on the regularization properties of the applied minimum-residual method. Particularly, we showed that the symmetric variants MINRES and MR-II do produce filtered SVD solutions, and in principle have properties similar to CGLS/LSQR. Contrary to CGLS/LSQR, the convergence of MINRES and MR-II and the particular solutions produced by these two methods, depend on the definiteness of the coefficient matrix – but this does not inhibit a regularizing effect of the iterations. On the other hand, GMRES and RRGMRES applied directly to a nonsymmetric problem do not in general produce regularized solutions. The underlying Krylov subspaces mix the desired solution components with the components that are contaminated by noise, and the convergence of the iterates can therefore be erratic. While both methods indeed minimize the residual norm in each iteration, it was also illustrated that this property in general does not correlate with the quality of the produced solutions. We can therefore imagine the following three cases when GMRES or RRGMRES is applied to a discrete ill-posed problem:

- a good regularized solution is produced and identified.
- a good regularized solution is maybe produced, but a stopping criterion does not detect this (in fact we may never know that we found a good solution).
- a good regularized solution cannot be produced at all from the provided solution subspace.

In practise this means that one should be very careful indeed to use these methods as general regularization methods.

To the best of our knowledge, a deeper theoretical understanding of GMRES and similar methods has not yet emerged – and neither has the basis for providing a deep theoretical analysis of the regularization properties. The insight developed in

this thesis is therefore based on a combination of theory and a number of numerical examples. This together provide a valuable new insight into the algorithms.

In parallel with the studies mentioned above, we also developed an approach that generalizes the concept of smoothing norms to minimum-residual methods. The primary obstacle for using the standard smoothing norms with these methods is that the resulting coefficient matrices are not square. The proposed method deals elegantly with this difficulty, and furthermore, it results in a symmetric coefficient matrix whenever the original coefficient matrix is symmetric. For some problems this method was seen to be favorable compared to the well-known PCGLS method. However, a further study, in part based on the results for the Krylov methods in standard form, revealed that the resulting iterations are not directly connected to the solutions of a general-form problem. Thus despite promising results for some problems, one cannot in general expect that the computed solutions satisfy the desired smoothing norm constraint. The results for these algorithms are therefore analog to the ones mentioned above for standard GMRES and RRGMRES – i.e., we cannot expect always to get a good regularized solution.

## 9.3 M$\mathcal{OO}$Re Tools

Most of the numerical examples and underlying pieces of code that have been used working with this project are either based directly on M$\mathcal{OO}$Re Tools, or developed specifically in the M$\mathcal{OO}$Re Tools framework. While M$\mathcal{OO}$Re Tools has definitely facilitated a lot of efficient implementations, it has also at times made life difficult. The large amount of objects in the object-oriented structure have to fit each other, and an object is naturally implemented – and tested – with a certain number of applications in mind. Thus, in new combinations, the already implemented objects may behave in undesirable ways. Therefore, to implement new functionality into the toolbox, or on top of the toolbox, is not always easy. Furthermore, the toolbox runs in the MATLAB environment, which means that MATLAB adds another layer of complexity to the functionality. (In fact M$\mathcal{OO}$Re Tools is known to work with MATLAB 6.5 and MATLAB 7.3, but not with MATLAB 7.0–7.2 due to a technical problem with these MATLAB versions.)

Nevertheless, the present project work resulted in a large number of improvements of the M$\mathcal{OO}$Re Tools basics, as well as several new implementations. Moreover, all the numerical examples show that M$\mathcal{OO}$Re Tools indeed works in practise.

## 9.4 Other Issues

Image deblurring problems have been treated in some detail in the thesis. We analyzed the underlying structures of the coefficient matrices that arise when different kinds of boundary conditions were applied. The work with the multidimensional regularization matrices was initially meant as a part of the image deblurring study – or slightly more generally as a part of a study of two-dimensional reconstruction

problems. But due to interfacing with other projects – especially the development of GravMag Tools, a toolbox for three-dimensional gravitational and magnetic reconstruction problems – a need for multidimensional regularization matrices arose. The work resulted in a deeper study of stacked Kronecker products and especially factorizations of those. The new M$\mathcal{OO}$Re Tools class @StackDerivativeOperator is in turn based on these developments.

Last, but not least, a brand new idea for locating the corner of a discrete L-curve was proposed. The approach is an attempt to make an automatic evaluation of the overall behavior of the L-curve instead of getting caught by small irrelevant details. The final implementation of this method is now a part of the adaptive pruning algorithm which is also distributed with M$\mathcal{OO}$Re Tools.

## 9.5 Perspectives

We discussed above the main achievements of the thesis. But while this discussion concluded the thesis, it does not end the work with inverse problems.

The obtained insight is important for the further developments in the field of inverse problems. It is important to note that a general conclusion about general minimum-residual methods and their properties when applied to discrete ill-posed problems cannot be based on a sparse set of examples for which the methods work. The present developments also show that there is a need for a deeper theoretical understanding of these methods before general conclusions can be drawn.

Recent research tries to characterize the convergence behavior of GMRES for specific classes of matrices. E.g., J. Liesen and Z. Strakoš [62] study GMRES applied to tridiagonal Toeplitz matrices. By studying specific classes of problems, it might be possible to theoretically identify situations in which GMRES and similar general minimum-residual methods are able to produce regularized solutions. Looking into the class of normal matrices might be fruitful, but the situation is definitely complicated, as an example from the thesis shows. For this purpose, one could look into the worst-case behavior of GMRES for normal matrices [64] and on the departure from normality [60].

The general minimum-residual methods are expensive to use because they need to carry along an entire set of basis vectors for the underlying Krylov subspace of increasing dimensions. For problems where GMRES is known to work, it would be interesting to investigate cheaper, but nonoptimal methods – e.g., QMR, BiCG, or the restarted variant GMRES(m). Again, a class of test problems for this purpose could be image deblurring problems with slightly nonsymmetric PSF matrices.

For large-scale problems, and especially iterative regularization methods, there is still a lack of robust stopping criteria. The new method due to Hansen et al. [42] based on a statistical test on the residual could be a promising new way to go. Regarding the implementation of stopping criteria for iterative methods, M$\mathcal{OO}$Re Tools also lacks a modular formulation of these issues.

Finally, there are several additional interesting issues to study and work with.

General preconditioning of the iterative methods is interesting for inverse problems where we want fast convergence, but only to the wanted parts of the solution. Using a flexible variant of GMRES, e.g., FGMRES [85] by Y. Saad, one can precondition each iteration of GMRES with a different preconditioner, even with another iterative algorithm. With a better understanding of GMRES (e.g., for a specific class of problems) a flexible preconditioning might be a way to go. Also, efficient iterative methods for solving inverse problem using more robust norms than the 2-norm are definitely interesting. Development and maintenance of M$\mathcal{OO}$Re Tools is also a continuous work for the future.

With all these words, let us wind up this thesis and finish with a quote:

*"I think and think for months and years.*
*Ninety-nine times, the conclusion is false.*
*The hundredth time I am right."*
Albert Einstein

Appendix A

# M$\mathcal{OO}$Re Tools – A Case Study

This appendix illustrates the process of implementing new functionality to M$\mathcal{OO}$Re Tools and especially how the new implementations might require revisions and changes to existing code.

The case study describes how a regularization operator based on the stacked Kronecker products as covered by Theorem 5.3 and its generalization to higher dimensions can be implemented. Furthermore, we want to use the regularization operator in connection with the large-scale M$\mathcal{OO}$Re Tools test problem interpolate.

## A.1 The Regularization Operator

For convenience, we repeat here the general structure of the wanted regularization matrices, which is

$$L = \begin{pmatrix} I_{d_N} \otimes I_{d_{N-1}} \otimes I_{d_{N-2}} \otimes \cdots \otimes I_{d_2} \otimes I_{d_1} \\ L_{d_N} \otimes I_{d_{N-1}} \otimes I_{d_{N-2}} \otimes \cdots \otimes I_{d_2} \otimes I_{d_1} \\ I_{d_N} \otimes L_{d_{N-1}} \otimes I_{d_{N-2}} \otimes \cdots \otimes I_{d_2} \otimes I_{d_1} \\ \vdots \\ I_{d_N} \otimes I_{d_{N-1}} \otimes I_{d_{N-2}} \otimes \cdots \otimes I_{d_2} \otimes L_{d_1} \end{pmatrix},$$

where the first row consists solely of identity matrices of sizes corresponding to the $N$ dimensions, and the latter rows contain one $L$ matrix each, and all in distinct positions. We define the SVDs of the $L_{d_i}$ matrices $L_{d_i} = U_i \Sigma_i V_i^T$ and construct the orthogonal transform of $L$

$$L_D = D(X_{d_N} \otimes X_{d_{N-1}} \otimes \cdots \otimes X_{d_1})^T, \tag{A.1}$$

where $X_i = V_i$ or $X_i = I$ depending on whether the $i$th "Kronecker column" contains a regularization matrix $L_{d_i}$ or not. The diagonal matrix $D$ consists of sums of Kronecker products of the singular values. We want to represent the factorization from (A.1) in MOORe Tools, and furthermore we must be able to construct the pseudoinverse

$$L_D^\dagger = (X_{d_N} \otimes X_{d_{N-1}} \otimes \cdots \otimes X_{d_1}) D^\dagger. \tag{A.2}$$

We consider two possible implementation strategies.

### A.1.1 Implementation Using the Existing Framework

In the existing MOORe Tools framework it seems straightforward to represent $L_D$ by two objects – a @DiagonalOperator D and a @KroneckerProduct X. To take advantage of specialized implementations for Kronecker products of two and three dimensions respectively the implementation becomes a bit complicated, because @KroneckerProduct2Ds and @KroneckerProduct3Ds must be constructed explicitly. Also, the vector defining the @DiagonalOperator must be of the right type, e.g., the @DiagonalOperator can have a @Vector2D defining the diagonal such that multiplications with the @DiagonalOperator and a @Vector2D results in a @Vector2D and not a @Vector. Performing the multiplication of the @DiagonalOperator and the @KroneckerProduct objects results in an @OperatorProduct object $O$ because the two objects cannot be directly multiplied together (without spoiling the structures).

For a vector $v$ of the right type (@Vector, @Vector2D, etc.), the product $L_D v$ can be computed with a new vector of the right type as result. Also the multiplication $L_D^T v$ is well defined. But we need also to compute pseudoinverse solutions, i.e., we need $L_D^\dagger v$ and $L_D^{\dagger T} v$. In principle the pseudoinverse is simply given by (A.2), but unfortunately, the pseudoinverse of the @OperatorProduct $O$ is not obtainable and solving the system O\v in MATLAB does not lead to a pseudoinverse solution, but a divide by zero error. This is a very basic problem.

For an @OperatorProduct, the inverse is only defined if all terms are square. The basic philosophy is of course that $(AB)^{-1} = B^{-1}A^{-1}$ if and only if $A$ and $B$ are both square and invertible. The @DiagonalOperator $O$ is square, but *not* necessarily invertible due to the zeros corresponding on the diagonal corresponding to the nullity of the regularization matrix.

The implementation of the @DiagonalOperator could easily be modified to compute the pseudoinverse solution, simply by only inverting the nonzero diagonal elements. For this particular @OperatorProduct, the result would then be correct. But in general $(AB)^\dagger \neq B^\dagger A^\dagger$, and it is therefore safer in general to require square and invertible matrices for an @OperatorProduct solve to be well defined; else the computation could result in an erroneous answer.

We could also try to remove the zero rows of the @DiagonalOperator, but then inverting the resulting @OperatorProduct is stopped by the @OperatorProduct itself because inversion of non-square elements are not allowed (for the same reason as above). Another problem of removing the zero rows is that the size of the vector

is changed, and the diagonal of the @DiagonalOperator can no longer be given as a vector of the right type and size. That is, we can no longer store the vector in a @Vector2D or @Vector3D object, but only in a @Vector object. Therefore, we need to keep track of the type of the original vector elsewhere which would complicate the implementation.

In summary, implementing the factorization of the stacked Kronecker product structure in an object composed by already implemented M$\mathcal{OO}$Re Tools constructs is not possible.

### A.1.2 Implementation in a New Class

The other possibility is to construct an entirely new class that implements the factorization, keeps track of the dimensions, and computes the pseudoinverse solution when an instance of the class is used in a system solve. There is some more work in doing so because a number of basic functions must be implemented for the new class to work in the M$\mathcal{OO}$Re Tools framework. Furthermore, we need to do the actual implementation in the hierarchy of M$\mathcal{OO}$Re Tools and not at user level. On the other hand, we can implement any functionality that the basic classes do not. The resulting class is the @StackDerivativeOperator that consists of the following methods:

**StackDerivativeOperator.m** The constructor for the class. This method sets up the stacked Kronecker products, and optionally computes the factorization and the corresponding nullspace vectors.

**sub_applytovector.m** Defines multiplication of an instance of the object with some vector object. The function overloads the MATLAB operator *.

**sub_solve.m** Defines system solves. This method always returns the pseudoinverse solution if the operator is rank-deficient. The function overloads the MATLAB operator \.

**display.m** Shows sensible information about the object on the command line – including the size of the operator and the size of the nullspace.

**sub_size.m** Returns the size of the object.

**sub_getmatrix.m** Computes the matrix representation of the object. Due to the efficient internal structure, the object may represent a matrix that is huge. Note that this might result in an out of memory error when computing the matrix representation of the object.

Below we show the implementation of the constructor StackDerivativeOperator.m. The constructor checks the inputs and setup parameters, then it creates the stacked Kronecker products using @IdentityOperators and @SparseMatrix operators, it constructs an operator with columns spanning the combined nullspace, and it constructs the factorization if needed.

```
function [D, N] = StackDerivativeOperator(n, d, grid, eco, weights),
% StackDerivativeOperator
%
% <Synopsis>
%   [L, N] = StackDerivativeOperator(n, d)
%   [L, N] = StackDerivativeOperator(n, d, grid, pack, weights)
%
% <Description>
%   The StackDerivativeOperator is an enhanced version of getL and
%   creates discrete approximations to derivative operators along
%   multiple dimensions. E.g.,
%
%     L = [I; Lx; Lyy]
%
%   where I is the identity of correct dimensions, Lx is an
%   approximation to the first derivative operator in the x-
%   direction, and Lyy is an approximation to the second derivative
%   operator in the y-direction. I.e., || L x ||_2 is an
%   approximation to a Sobolev norm of x. In standard form, Kronecker
%   products are used to form the rows of L for problems of more than
%   one dimension. For two-norm regularization, and especially when
%   applying regularizing iterations, a more attractive form is
%
%     L_pack = D*V
%
%   where D is a diagonal matrix and V is a Kronecker product. It
%   holds that L_pack = Q*L where Q is orthogonal. I.e., when
%   working with two-norms || L x ||_2 = || L_pack x ||_2.
%
% <Input Arguments>
% * n     The dimensions of the problem given as a vector, e.g.,
%         n = [10 20 30] defines a three-dimensional problem of
%         size 10 x 20 x 30.
% * d     Cell array with derivatives to use. First index is a
%         boolean for including the std. two-norm or not. Next
%         length(n) elements define the derivatives to use for the
%         different dimensions. E.g. d = {1, [], 1, 2} uses ||x||
%         plus first derivative along the second dimension and the
%         second derivative along the third dimension. The first
%         entry must be '1' or [], whereas the last entries can
%         also be cell arrays that implement special operators and
%         corresponding nullspaces. E.g.,
%
%              d = {[], {Lx, Nx}, {Ly, []}, []};
%
%         use the LinearOperators Lx and Ly along the two first
%         dimensions. The operator Lx has a nullspace spanned by
%         the Vectors of the VectorCollection Nx, and Ly has no
%         nullspace. No operator (apart from the identity) is
%         applied to the third dimension. Ls and Ns must have a
%         form corresponding e.g., to the ones returned from getL.
% * grid  A struct defining grid spacings different from 1. E.g.,
%         grids = {[1 4 5], [4 4 4 4]} defines the grid spacing for a
%         problem of dimensions n = [4 5]. Default is a grid spacing
%         of all ones, and the argument can be left out by []. For
```

```
56  %         dimensions with non-uniform grids, d must be 0, 1, or 2.
57  % * pack  Compute and use the orthogonal transformation of L instead
58  %         of L. I.e., return L_pack instead of L.
59  % * w     A vector containing weights for each row of the stacked
60  %         Kronecker products. It must hold that length(w) equals
61  %         the number of non-empty cells in d.
62  %
63  % <See Also>
64  %   WeightedPseudoInverse, getL, getLgrid
65
66  % Toke Koldborg Jensen, IMM, DTU, 2005
67
68  % Last revised $Date: 2005/10/21 11:33:11 $ by $Author: tkj $
69  % $Revision: 1.3 $
70
71  % Check input arguments and setup
72  switch(nargin)
73   case 0
74    D.V = [];
75    D.D = [];
76    D.N = [];
77    D.eco = [];
78   case 1
79    if isa(n, 'StackDerivativeOperator'),
80      D = n;
81      return
82    else
83      error('At least two input arguments, please');
84    end
85   otherwise
86    dim = length(n);
87
88    if length(d)~=dim+1
89      error(sprintf('Length of second argument should be length(n)+1 = %d', dim+1));
90    end
91
92    % Extract cell array to a boolean list of indices to derivative operators
93    for i=1:length(d),
94      dtmp(i) = ~isempty(d{i});
95    end
96    numStack = sum(dtmp>0);
97    [dummy, dIndex] = find(dtmp(2:end)>0);
98
99    if nargin<3 | isempty(grid),
100     grid = cell(1,length(n));
101     for i=1:dim,
102       grid{i} = ones(1,n(i)-1);
103     end
104   end
105
106   if nargin<4 | isempty(eco),
107     if numStack>1
108       eco = 1;
109     else
110       eco = 0;
```

```
111       end
112     end
113
114     if nargin<5,
115       weights = ones(1,numStack);
116     end
117
118     if length(weights)~=numStack,
119       error(['Number of weights must correspond to number of terms in ' ...
120              'd different from 0']);
121     end
122
123     weights = exp(log(weights)/dim); % multiply each term of
124                                      % kronecker products with
125                                      % weight! I.e, input = weights.^dim
126
127     if numStack>1, % if more than one row, then initialize OperatorArray
128       L = OperatorArray(numStack,1);
129     end
130
131     N = cell(1,dim);
132     for i=1:numStack,
133       if i==1 & ~isempty(d{1}), % Include standard norm -- no derivative
134         for j=1:dim,
135           KP{j} = weights(i)*IdentityOperator(n(dim-j+1));
136           N{j} = Matrix(zeros(n(dim-j+1),0));
137         end
138       else
139         dimension = dIndex(i-~isempty(d{1}));
140         dimLocation = dim-dimension+1;    % in Kron... opposite order
141         for j=1:dim,
142           if isempty(N{j}), N{j} = IdentityOperator(n(dim-j+1)); end;
143           if j==dimLocation,
144             if prod(size(d{dimension+1}))==0,  % No entry --> use identity
145               KP{j} = IdentityOperator(n(dimension));
146               W = Matrix(zeros(n(dimension),0));
147             elseif prod(size(d{dimension+1}))==1,
148               [KP{j}, W] = getLgrid(n(dimension), d{dimension+1}, ...
149                                                   grid{dimension});
150             else
151               CELL = d{dimension+1};
152               KP{j} = CELL{1}; W = CELL{2};
153               if size(KP{j},2)~=n(dimension),
154                 error('Wrong dimensions of input object - L');
155               end
156               if isempty(W),
157                 W = Matrix(zeros(n(dimension),0));
158               elseif size(W,1)~=n(dimension),
159                 error('Wrong dimensions of nullspace object - N');
160               end
161             end
162             KP{j} = weights(i)*KP{j};
163             if isempty(d{1})
164               N{j} = W;
165             end
```

```
166           else
167             KP{j} = weights(i)*IdentityOperator(n(dim-j+1));
168           end
169         end
170       end
171
172       switch dim,
173        case 1
174         L{i,1} = KP{1};
175        case 2
176         L{i,1} = KroneckerProduct2D(KP{1}, KP{2});
177        case 3
178         L{i,1} = KroneckerProduct3D(KP{1}, KP{2}, KP{3});
179        otherwise
180         L{i,1} = KroneckerProduct(KP);
181       end
182     end
183
184     switch dim
185      case 1
186       N = N{1};
187      case 2
188       v = Vector2D(zeros(size(N{2},2), size(N{1},2)));
189       N = KroneckerProduct2D(N{1},N{2});
190      case 3
191       v = Vector3D(zeros(size(N{3},2), size(N{2},2), size(N{1},2)));
192       N = KroneckerProduct3D(N{1},N{2},N{3});
193      case 4
194       dimnull = [];
195       for i=1:size(N,2), dimnull = [size(N{i},2) dimnull]; end
196       v = VectorND(zeros(dimnull));
197       N = KroneckerProduct(N);
198     end
199
200     % Extract Kronecker Product notation to VectorCollection
201     W = VectorCollection(size(N,2));
202     for i=1:size(N,2),
203       v(i) = 1;
204       W(:,i) = N*v;
205       v(i) = 0;
206     end
207     N = W;
208
209     for i=1:dim
210       S{i} = IdentityOperator(n(dim-i+1));
211       V{i} = IdentityOperator(n(dim-i+1));
212     end
213
214
215     % Create smart representation to system solves
216     if eco~=0 % & numStack>1,
217       for i=1:numStack,
218         for j=1:dim,
219           if dim>1,
220             TERM = get(L{i,1}, j);
```

```
221           else
222             TERM = L{i,1};
223           end
224           if ~isa(TERM, 'IdentityOperator')
225             [U{j}, S{j}, V{j}] = svd(full(getmatrix(TERM)));
226             V{j} = Matrix(V{j});
227             Ssq = diag(S{j}'*S{j}); % squared singular values
228             KP{j} = Matrix(Ssq);
229           else
230             KP{j} = Matrix(ones(n(dim-j+1),1));
231           end
232         end
233         dg = KroneckerProduct(KP);
234         dia(:,i) = getmatrix(dg);
235       end
236       dia = sqrt(sum(dia,2));
237       switch dim
238        case 3
239         D.V = KroneckerProduct3D(V{1}, V{2}, V{3});
240         diagV = Vector3D(reshape(dia, n(1), n(2), n(3)));
241        case 2
242         D.V = KroneckerProduct2D(V{1}, V{2});
243         diagV = Vector2D(reshape(dia, n(1), n(2)));
244        case 1
245         D.V = V{1};
246         diagV = Vector(dia);
247        otherwise
248         D.V = KroneckerProduct(V);
249         for i=1:dim, nd{i} = n(i); end
250         diagV = VectorND(reshape(dia, nd{:}));
251       end
252       D.D = DiagonalOperator(diagV);
253     else
254       if prod(size(L))==1,
255         D.V = L{1,1};
256       else
257         D.V = L;
258       end
259       D.D = [];
260       eco = 0;
261     end
262
263     D.N = N;
264     D.eco = eco;
265   end
266
267   D = class(D, 'StackDerivativeOperator', LinearOperator);
268   superiorto('Vector');
```

Apart from help texts and initialization, we see that the code includes several branches for constructing specific objects for one, two, three, and general dimensions. These are needed to take advantage of the specialized classes for the various dimensions. Thus the implementation heavily depends on the already implemented MOORe Tools classes. Note that the operator that defines the nullspace vectors is

a @VectorCollection – which is nothing but a collection of vector objects (possibly of different type). The nullspace representation is generated in the lines 184–207. To generate the nullspace from the one-dimensional nullspace objects (that apply to each separate dimension), the easiest way is to use a @KroneckerProduct of the right type (lines 184–198). But for using the nullspace operator afterwards, the @VectorCollection provides a more useful representation. This is because a @VectorCollection can in principle contain user defined vector types, whereas a @KroneckerProduct object can only be applied to vectors of the correct type.

Unfortunately, the @VectorCollection representation was problematic for higher dimensions. E.g., if the problem is four dimensional and the nullspace is spanned by one vector only, then dimnull=[1 1 1 1] (in line 195) and we want to construct v as a VectorND of dimension four, but having only one element. Such a vector, when constructed, is returned as a @VectorND of only 2 dimensions containing one element. Even though the vector is generated in the right way, it is not compatible with a @KroneckerProduct object of four dimensions. Several other problems appeared with the inner workings of the @VectorCollection functions, and a lot of bug fixes (some connected to these issues) are described in Appendix B.

A @StackDerivativeOperator object contains the following fields

**D.V** When no factorization is used, this variable contains the ordinary stacked Kronecker products and is of type @OperatorArray. When the factorization is used, D.V contains the Kronecker product $(X_{d_N} \otimes X_{d_{N-1}} \otimes \cdots \otimes X_{d_1})$ from (A.1).

**D.D** The diagonal matrix of the factorization. When no factorization is used, then D.D is empty.

**D.N** A @VectorCollection with columns that span the nullspace of $L$.

**D.eco** A boolean value ("economy") that determines whether the factorization is used or not.

**D.LinearOperator** The parent class.

We will not go into detail with all the implemented methods, but just show how the sub_solve.m method is implemented, such that a system solve will result in the pseudoinverse solution.

```
1  function v = sub_solve(D, v),
2  % sub_solve Solve linear equation
3  %
4  % <Description>
5  %   Called by LinearOperator/mldivide who takes care of scaling. Calculates
6  %   the Moore-Penrose pseudoinverse solution to the system using the possibly
7  %   rank-deficient (factorization of) stacked Kronecker products.
8  %
9  % <See Also>
10 %   LinearOperator/mldivide
11
12 % Toke Koldborg Jensen, IMM, DTU, 2005.
```

```
13
14  % Last revised $Date: 2005/09/15 12:34:03 $ by $Author: tkj $
15  % $Revision: 1.1 $
16
17  if gettransposed(D),
18    if D.eco==0 % No factorization is used
19      v = D.V'\v;
20    else
21      dia = diag(D.D);
22      data = getdata(dia);
23      I = find(data==0); data(I) = 1;
24      dia = setdata(dia, data);
25      v = dia.\(D.V'*v);
26      v(I) = 0;
27    end
28  else
29    if D.eco==0 % no factorization is used
30      v = D.V\v;
31    else
32      dia = diag(D.D);
33      data = getdata(dia);
34      I = find(data==0); data(I) = 1;
35      dia = setdata(dia, data);
36      v = dia.\v; v(I) = 0;
37      v = D.V*v;
38    end
39  end
```

We see that if the factorization is not used (D.eco==0), then the ordinary stacked Kronecker products (stored in the @OperatorArray D.V) are used directly by forwarding the solve to the @OperatorArray/sub_applytovector.m. If the factorization is used, then the pseudoinverse of the @DiagonalOperator is computed explicitly by only inverting the nonzero elements. The implementation is complicated a little by the need for retaining the correct vector type all the way through the computations. That is, zero elements are set to one, and the elementwise divisions v = dia.\(D.V'*v); and v = dia.\v; are performed using the correct objects. Finally, the elements that would have caused a divide-by-zero error are set manually to zero and kept in the resulting vector to be able to store it as a vector of the same dimensions.

Now, we can construct objects of the @StackDerivativeOperator type, which have the desired functionality.

### A.1.3 The Interpolation Problem

We want to use the @StackDerivativeOperator in connection with the MOORe Tools test problem interpolate (used in Examples 4.14 and 5.5). This test problem is different from most of the other test problems because it relies on specialized classes as well as interfacing to mex-files. To represent measurements on an irregular grid, the class @GridVector used. And even though the domain described is two-dimensional, a @GridVector is different from a @Vector2D. The operator used with the interpolate test problem uses @GridVectors as both input and output, and the matrix-vector

multiplications are implemented in external C-functions.

Imagine that we create a test problem and add noise to the right-hand side. The regular grid is $20 \times 20$, and we create 200 irregularly spaced measurement points. We also create an instance of the newly developed class

```
[A,bex,x] = interpolate(20, 200, 100);
e = randn(size(bex));
e = e/norm(e)*norm(bex)*1e-2;
b = bex + e;
[L,N] = StackDerivativeOperator([20, 20], {[], 2, 2});
```

But now we run into trouble. A two-dimensional @StackDerivativeOperator can be applied to @Vector2Ds which are compatible with the internal @KroneckerProduct2Ds. But using L with @GridVectors is not defined. To circumvent this difficulty, we could extract the vectors from the @GridVectors and store them in normal @Vectors – now without information about irregular sampling points or regular grids. We therefore loose the possibility to correctly illustrate the irregularly placed measurements. Similarly, we can construct the matrix representations of the operator A and L, but for larger problems we risk to run out of memory, and we definitely do not use the clever mex-implementation for the interpolation.

More elegantly, we would like to work with @GridVectors whenever needed, and be able to represent the regularly spaced solutions as @Vector2Ds when necessary. This cannot easily be done on user level, and therefore we created a new auxiliary class called @GridVectorReshape. The functionality of this class is similar to the basic M$\mathcal{OO}$Re Tools class @VectorReshape that acts like an identity matrix, which transforms one basic vector type into another. The @GridVectorReshape contains information about the grids through an internally stored @GridVector. The following code generates a @GridVector in the solution domain A'*b and uses this to construct a @GridVectorReshape object.

```
Vr = GridVectorReshape(A'*b);
L = L*Vr;
```

Now, L'\x returns a @Vector2D, and L\(L'\x) returns a @GridVector. To make all this work smoothly, a few functions, such as sub_size.m and sub_solve.m are also implemented for the @GridVectorReshape class. If the sub_solve.m method is not implemented, then the @OperatorProduct L*Vr cannot be constructed due to an apparent mismatch of the inner dimensions because the size of the @GridVectorReshape object will appear to be $0 \times 0$.

The next problem is that in the pcgls.m method, we need to compute a thin QR factorization of the product AN, and now we have the opposite problem. The interpolate object A cannot be applied to the @Vector2Ds in N. Again, we need to use the @GridVectorReshape

```
N = Vr'*N;
```

which results in a @VectorCollection consisting of @GridVectors. Note that if the nullspace operator N was stored as a @KroneckerProduct2D then the above reshaping of the vectors would not have been possible (see the discussion on the representation of the nullspace vectors in the previous section).

Now the product A*N results in another @VectorCollection of @GridVectors for which the thin QR factorization is in principle defined. Unfortunately, another error occurred when computing the QR factorization because the @GridVector class did not implement a subsasgn.m function. Instead it used the @Vector/subsasgn.m through inheritance and the error occurred because the data of the @GridVector class is not stored in the internal field data which is the one that is used in @Vector/subsasgn.m. To solve this final problem, we implemented a specialized function subsasgn.m for the existing @GridVector class such that the correct field is updated.

With all the described changes and modifications (and several more), we can finally run pcgls.m on an Interpolate test problem combined with a smoothing norm defined through a @StackDerivativeOperator and a @VectorCollection containing @GridVectors that span the corresponding nullspace.

## A.2 Use with GravMag Tools

GravMag Tools is another object-oriented toolbox developed at IMM, DTU by MSc Jesper Pedersen, and it is specialized to solve gravitation and magnetization problems. The classes of this toolbox are all inherited from MOORe Tools, and therefore all the basic functionality and all the algorithms implemented in MOORe Tools are directly available to GravMag Tools.

As seen from the last section, even interfacing a simple test problem with a more general MOORe Tools construct can be difficult. Thus building an entire toolbox is not done without problems. During the development of GravMag Tools a lot of problematic issues with MOORe Tools have been identified. Similarly, the development of GravMag Tools has influenced a great deal on the developed extensions to MOORe Tools.

## A.3 Summary

In this appendix, we saw that implementing new functionality in the object-oriented framework of MOORe Tools is not simple and straightforward. All objects rely on the functionality of the other objects, and even though the general functionality of a class is well tested, one might run into problems if more specific functionality is needed.

Future development of specialized code on top of MOORe Tools will hopefully benefit from the large amount of corrections, changes and modifications done to MOORe Tools as a result of the close interplay between the development of this thesis, GravMag Tools, and MOORe Tools itself. Furthermore, this chapter hopefully

serves as an illustration to future developers of some of the problems one might run into during the development phase.

One final remark – initially, consider the functionality of M$\mathcal{OO}$Re Tools to be correct. But if debugging own code does not remove the errors, then it is probable that you have come across some problem with M$\mathcal{OO}$Re Tools. Do not hesitate to look in the code.

Appendix B

# List of M$\mathcal{OO}$Re Tools Changes

This appendix shows a list of the most important changes, corrections and additions done to M$\mathcal{OO}$Re Tools. It may, in connection with the previous Appendix, serve as a guideline for future developers who want insight into how M$\mathcal{OO}$Re Tools work.

- Added: @SVDOperator/picard.m. A function that generates Picard plots based on M$\mathcal{OO}$Re Tools objects.
- Added: @InverseOperator class. Makes it possible to include also inverses and pseudoinverses in @OperatorProducts. Basically, the class switches the meaning of the MATLAB operators * and \. The class includes only the most necessary methods.
- Added: @StackDerivativeOperator. This class is described in the thesis.
- Added: @LinearOperator/pcgls.m. Added the M$\mathcal{OO}$Re Tools implementation of PCGLS.
- Added: Algorithms/corner.m. The new adaptive pruning algorithm for locating the corner of a discrete L-curve is included as an auxiliary function.
- Added: Algorithms/progress.m. Support for progress bars for iterative algorithms.
- Added: Algorithms/getLgrid.m. A special getL.m that computes discrete approximations to first and second derivative operators as well as their nullspaces based on a non-equidistant grid.

- Added: @CirculantOperator. This class represents circulant matrices efficiently and performs all matrix-vector multiplications through FFTs. It also implements eig.m for returning the eigenvalues.
- Added: @CirculantOperator2D. A class for handling block circulant matrices with circulant blocks. Two-dimensional FFTs are used for the matrix-vector multiplications.
- Added: @Vector/prod.m. Returns the product of all elements in a vector. This functionality was needed when implementing the picard.m function and now implemented in general.
- Added: TestExamples/@GridVector/getcoords.m. This function returns the coordinates of the points in the @GridVector such that it is possible to make other plots than the ones provided by the @GridVector itself.
- Added: Support for progress bars implemented in all iterative methods.
- Fix: @KroneckerProduct3D/sub_applytovector.m. The dimensions in the multiplication were wrong. The M$\mathcal{OO}$Re Tools test routine testkron3d was so unfortunate not to catch the error. The multiplication is corrected!
- Fix: @LinearOperator/cgls.m. The reorthogonalization part is corrected. An error in the indices meant that the first search direction was left out of the reorthogonalization procedure.
- Fix: LinearOperator/minres_mt.m. The implementation performed two matrix-vector multiplications pr. iteration. It has been changed and the result of $Av_1$ is now stored in a temporary variable to avoid multiple multiplications!
- Fix: @LinearOperator/mrdivide.m. The return value was never set, and the implementation did not work. It called solve (from symbolic toolbox) instead of sub_solve from the appropriate class. Furthermore, A and B both first and third branch have been switched.
- Change: @LinearOperator/mtimes.m. If one of the inputs is a double matrix, then it is transformed into a @Matrix object. When this is not done, then the scaling parameter of the @LinearOperator is set to a double matrix, which results in an unexpected behavior. With the new implementation, the operatormtimes.m is called instead.
- Fix: @KroneckerProduct/sub_operatormtimes.m. The size of the resulting object was not updated – i.e., even though two Kronecker products of different sizes are multiplied together, then the resulting object still returned the sizes of the original first object. The sizes are now updated.
- Fix: @KroneckerProduct/sub_solve.m. The last help text is corrected. A ' was missing, and objects by them self do not implement sprintf which resulted in an error.

- Fix: @KroneckerProduct/sub_applytovector.m. Multiplication with the transpose operator was not defined.

- Change: @Matrix/sub_operatormtimes.m. A check of the sizes here is not necessary because this check is already performed in general by the @LinearOperator/mtimes.m. The size check is removed.

- Issue: @IdentityOperators can in principle represent a diagonal that is not all ones. This funny behavior appears because the scaling parameter resides in the super class @LinearOperator and therefore an @IdentityOperator can also be scaled. Moreover, multiplying an @IdentityOperator with another object should in principle return just the other object and not an @OperatorProduct which is often the case because the parent of the @IdentityOperator (the @DiagonalOperator) might not support multiplication with the other object. This can be fixed in the @LinearOperator/mtimes.m such that the multiplication simply returns the other object and scales the constant. But for the special @VectorReshape versions of the @IdentityOperator, the explicit construction of the @OperatorProduct is desirable. Several changes to deal with these issues have been skipped again.

- Fix: @LinearOperator/lsqr_mt.m. The default tolerance on the residual was set to 1e-6norm(b) and not 1e-6norm(A'*b).

- Change: @VectorCollection/subsref.m. It was not possible to extract single elements and subvectors from a @VectorCollection. E.g., if a Kronecker product is made up of @VectorCollections then @KroneckerProduct/sub_applytovector.m does not work because it needs to extract the elements of the @VectorCollections one by one. The method @VectorCollection/subsref.mis extended such that sub vector collections and single elements can be extracted.

- Issue: Vectors do not behave identically. If one element is extracted from a @Vector or @Vector3D then a double is returned, but if one element is extracted from a @Vector2D or @VectorND then a @Vector object of size one is returned! This is not consistent. In some occasions the first behavior is preferred, and in other occasions the other. No changes are made – but be careful what to expect.

- Fix: @Vector3D/subsasgn.m. Error in the logical expression s.subs1==':'. This is unfortunately also true for s.subs1==58 because the ASCII value of ':' is used. This logical expression now used a strcmp instead.

- Change: @VectorND. There were problems when creating a vector of more than three dimensions with VectorND when the multidimensional vector is degenerate such that the two last dimensions have size one. In this case, the resulting VectorND will appear to have dimension 2! Even though the actual size of the internal data is correct, it gives problems when objects checks

for the size of the @VectorND and gets a wrong answer. To fix the problem, one needs to specify the exact dimension of the vector, and to do this, the following functions are changed: @VectorND/VectorND.m, @VectorND/size.m, @VectorND/display.m, and @KroneckerProduct/sub_applytovector.m. In the latter we have to consider explicitly the situation where the size of the 3rd, 4th, etc. dimension have size one. Similar change was needed in @KroneckerProduct/sub_solve.m.

- Fix: @IdentityOperator/subsref.m. When only one element is wanted, e.g., if I is an @IdentityOperator, elmt = I(2,2);. Here, a reference to the field A.v does not exist. The implementation has been corrected by extracting the vector through a v = diag(A);.

- Fix: @OperatorProduct/subsasgn.m. A size was missing!

- Issue: @Vector/pnorm.m. Norms $\|\cdot\|_p$ with $p < 2$ are not smooth at zero, and therefore the second derivative here is infinite. This spoils the convergence of Newton-like methods like the implemented @LinearOperator/gmin.m. On one hand, @Vector/pnorm.m should return the correct p-norm, but on the other hand, one could for practical reasons add a small fudge parameter such that the second derivative is only huge in zero, but not infinite.

- Issue: @KroneckerProduct2D/gsvd.m. This function is not complete and has not been changed.

- Fix: @VectorND/size.m. The second branch of the if sentence should be nargin==2 and not nargin==1.

- Fix: @KroneckerProduct3D/sub_solve.m. The dimensions of the returned @Vector3D are ordered oddly. Especially, if L is a @KroneckerProduct3D and v is a corresponding @Vector3D such that V'\v is well defined then V\(V'\v) fails. If @KroneckerProduct and @VectorND are used instead, then the dimensions are correct! The reshapes in @KroneckerProduct3D/sub_solve.m are changed such that the result is correct.

- Fix: @OperatorArray/sub_applytovector.m. MATLAB7.3 complained when assigning a struct to a nonstruct variable. The problem is fixed by introducing a temporary variable vv in the construction of the VectorStack (lines 37–38).

- Fix: @OperatorArray/OperatorArray.m. Order of fields in the first branch of the switch case are switched to appear in the same order as in the second branch. It gave problems when reloading a saved object.

- Change: @Vector2D/subsref.m. Added code such that a sub Vector2D can be returned. E.g., if x=Vector2D(randn(10)) then x(1:2,1:2) is a Vector2D consisting of the first $2 \times 2$ elements of the original Vector2D. The colon notation also

applies, e.g., x(:,2:3). (This change is needed e.g., in connection with the implementation of embedded Toeplitz+Hankel matrices in larger block circulant matrices.)

- Fix: @LinearOperator/lbhybrid.m. Some errors have been corrected. 1) When computing the solution iterate x, the entire VectorCollection V was used. In case more than one iterate should be returned, this resulted in a dimension mismatch because also empty vectors in a VectorCollection are included in the size of it. I.e., V was changed to V(:,1:i). 2) The first return value was set to x and not X, i.e., only the last calculated solution iterate was returned, and not the wanted VectorCollection X. Support for progress bars has also been implemented.

- Fix: @SVDOperator/l_corner. The L-curve implementation in M$\mathcal{OO}$Re Tools is based directly on the L-curve implementation from Regularization Tools. But in Regularization Tools the calculation of the SVD is done using the csvd from Regularization Tools i.e., the compact SVD whereas in M$\mathcal{OO}$Re Tools the full SVD is used. This sometimes lead to a mismatch of the dimensions. If e.g., the coefficient matrix $A \in \mathbb{R}^{m\times n}$ is such that $m > n$, then the full SVD $A = U\Sigma V^T$ has $U \in \mathbb{R}^{m\times m}$ whereas the compact SVD $A = \overline{U\Sigma V}^T$ has $\overline{U} \in \mathbb{R}^{n\times n}$. Line 67 of @SVDOperator/l_corner.m is changed from beta = U'*b; to beta = U(:,1:p)'*b;.

- Change: @Vector/mtimes.m. In the original implementation, no outer products were supported. The implementation is changed such that if one of the input vectors has length one, then the "outer product" between the two is supported – in fact this is just a scalar times vector. The functionality is needed, e.g., to compute a TSVD solution with one component only.

- Fix: @Matrix/sub_svd.m. If the input matrix has only one row or one column, then S = DiagonalOperator(Vector(diag(S)), rows, cols); gives an error because diag(S) then becomes a matrix, which is incompatible with the @Vector constructor. A check of the dimensions has been implemented to support also this case.

Appendix C

# Included Papers

This appendix lists for convenience the papers that have been written during this PhD study. Included are:

- "An Adaptive Pruning Algorithm for the Discrete L-curve Criterion". The version provided here is the latest revision which has been accepted for publication in Journal of Computational and Applied Mathematics. Note that the type setting of the final publication may differ from the one presented here. The paper is co-authored by P. C. Hansen, T. K. Jensen, and G. Rodriguez.
- "Smoothing-Norm Preconditioning for Minimum-Residual Methods". The manuscript includes the final revisions and is accepted for publication in SIAM Journal on Matrix Analysis and Applications. Final layout may differ. The paper is co-authored by P. C. Hansen and T. K. Jensen.
- "Iterative Regularization with Minimum-Residual Methods". The paper is shown as submitted to BIT and is co-authored by T. K. Jensen and P. C. Hansen.

# An Adaptive Pruning Algorithm for the Discrete L-Curve Criterion ⋆

Per Christian Hansen *, Toke Koldborg Jensen

*Informatics and Mathematical Modelling, Building 321*
*Technical University of Denmark, DK-2800 Lyngby, Denmark*

Giuseppe Rodriguez

*Dipartimento di Matematica e Informatica, Università di Cagliari*
*viale Merello 92, I-09123 Cagliari, Italy*

**Abstract**

We describe a robust and adaptive implementation of the L-curve criterion. The algorithm locates the corner of a discrete L-curve which is a log-log plot of corresponding residual norms and solution norms of regularized solutions from a method with a discrete regularization parameter (such as truncated SVD or regularizing CG iterations). Our algorithm needs no predefined parameters, and in order to capture the global features of the curve in an adaptive fashion, we use a sequence of pruned L-curves that correspond to considering the curves at different scales. We compare our new algorithm to existing algorithms and demonstrate its robustness by numerical examples.

*Key words:* Discrete ill-posed problems, L-curve criterion, regularization, parameter-choice method.

⋆ This work was supported in part by grant no. 21-03-0574 from the Danish Natural Science Research Foundation, and by COFIN grant no. 2002014121 from MIUR (Italy).

* Corresponding author
*Email addresses:* pch@imm.dtu.dk (Per Christian Hansen), toke.jensen@gmail.com (Toke Koldborg Jensen), rodriguez@unica.it (Giuseppe Rodriguez).
*URLs:* http://www.imm.dtu.dk/~pch (Per Christian Hansen), http://bugs.unica.it/~gppe/ (Giuseppe Rodriguez).

## 1 Introduction

We are concerned with discrete ill-posed problems, i.e., linear systems of equations $A\,x = b$ or linear least squares problems $\min \|A\,x - b\|_2$ with a very ill-conditioned coefficient matrix $A$, obtained from the discretization of an ill-posed problem, such as a Fredholm integral equation of the first kind. To compute stable solutions to such systems under the influence of data noise, one must use regularization. The amount of regularization is controlled by a parameter, and in most cases it is necessary to choose this parameter from the given problem and the given set of data.

In this paper we consider regularization methods for which the regularization parameter takes discrete values $k$, e.g., when the regularized solution lies in a $k$-dimensional subspace. Examples of such methods are truncated SVD and regularizing CG iterations. These methods produce a sequence of $p$ regularized solutions $x_k$ for $k = 1, 2, \ldots, p$, and the goal is to choose the optimal value of the parameter $k$. A variety of methods have been proposed for this parameter choice problem, such as the discrepancy principle, error-estimation methods, generalized cross-validation, and the L-curve criterion. For an overview, see Chapter 7 in [6].

This work focuses on the L-curve criterion, which is based on a log-log plot of corresponding values of the residual and solution norms:

$$(\,\log \|A\,x_k - b\|_2\,,\ \log \|x_k\|_2\,), \qquad k = 1, \ldots, p. \tag{1}$$

For many problems arising in a variety of applications, it is found that this curve has a particular "L" shape, and that the optimal regularization parameter corresponds to a point on the curve near the "corner" of the L-shaped region; see, e.g., [6, §7.5] or [7] for an analysis of this phenomenon.

For continuous L-curves it was suggested in [8] to define the corner as the point with maximum curvature. For discrete L-curves it is less obvious how to make an operational definition of a corner suited for computer implementation. While a few attempts have been made [2], [5], [9], we feel that there is a need for an efficient and robust general-purpose algorithm for computing the corner of a discrete L-curve. The algorithm developed in this paper achieves its robustness through a combination of several adaptive strategies, and the Matlab codes is available from the authors.

The L-curve criterion has its limitations; it does not work well when the solution is very smooth [3], and the regularization parameter may not behave consistently with the optimal parameter as the problem size goes to infinity [10] (see also the discussion of these limitations in [7]). It is still worthwhile to have access to a robust algorithm for discrete L-curves, partly because the

method can work very well, and partly because it can be used together with other methods with other limitations.

Our paper is organized as follows. Section two introduces the discrete L-curve and summarizes previous algorithms for finding the corner. Section three is the main contribution of the paper and describes the new algorithm in detail. The algorithm is tested in section four where the performance is shown using a series of smaller test problems as well as a large-scale problem.

## 2 The Discrete L-Curve Criterion

For convenience assume that the coefficient matrix $A$ is of dimensions $m \times n$ with $m \geq n$, and let the SVD of $A$ be given by $A = \sum_{i=1}^{n} u_i \, \sigma_i \, v_i^T$. What characterizes a discrete ill-posed problem is that the singular values $\sigma_i$ decay gradually to zero, and that the absolute value of the right-hand side coefficients $u_i^T b$ from some point, and on the average, decay faster.

The best known regularization method with a discrete regularization parameter is probably the truncated SVD (TSVD) method. For any $k < n$ the TSVD solution $x_k$ is defined by

$$x_k = \sum_{i=1}^{k} \frac{u_i^T b}{\sigma_i} \, v_i. \tag{2}$$

The residual and solution norms for $x_k$ are given by

$$\|A\, x_k - b\|_2^2 = \sum_{i=k+1}^{n} (u_i^T b)^2, \qquad \|x_k\|_2^2 = \sum_{i=1}^{k} \left( \frac{u_i^T b}{\sigma_i} \right)^2. \tag{3}$$

The CGLS algorithm is mathematically equivalent to applying the CG method to the normal equations, and when applied to ill-posed problems this method exhibits semi-convergence, i.e., initially the iterates approach the exact solution while at later stages they deviate from it again. Moreover, it is found that the number $k$ of iterations plays the role of the regularization parameter. Hence, these so-called regularizing CG iterations also lead to a discrete L-curve. For more details see, e.g., §§6.3–6.5 in [6].

In the rest of the paper we occasionally need to talk about the oriented angle between the two line segments associated with a triple of L-curve points, with the usual convention of the sign of the angle. Specifically, let $\mathcal{P}_j$, $\mathcal{P}_k$ and $\mathcal{P}_\ell$ be three points satisfying $j < k < \ell$, and let $v_{r,s}$ denote the vector from $\mathcal{P}_r$ to $\mathcal{P}_s$. Then we define the oriented angle $\theta(j,k,\ell) \in [-\pi, \pi]$ associated with the triplet as the angle between the two vectors $v_{j,k}$ and $v_{k,\ell}$, i.e.,

$$\theta(j,k,\ell) = \angle(v_{j,k}, v_{k,\ell}). \tag{4}$$

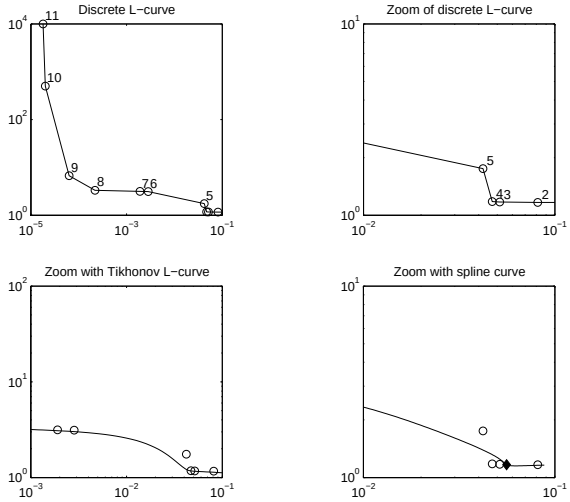



Fig. 1. Top: a discrete L-curve for TSVD with a global corner at $k = 9$ and a little "step" at $k = 4$; the smallest angle between neighboring triplets of points occurs at $k = 4$. Bottom left: part of the Tikhonov L-curve for the same problem. Bottom right: part of the 2D spline curve used by the Matlab function l_corner in Regularization Tools [5]; the point on the spline curve with maximum curvature is indicated by the diamond.

With this definition, an angle $\theta(j, k, \ell) < 0$ corresponds to a point which is a potential candidate for the corner point, while $\theta(j, k, \ell) \geq 0$ indicates a point of no interest.

In principle, it ought to be easy to find the corner of a discrete L-curve: compute all angles $\theta(k-1, k, k+1)$ for $k = 2, \ldots, p-1$ and associate the corner point $\mathcal{P}_k$ with the angle closest to $-\pi/2$. Unfortunately, this simple approach is not guaranteed to work in practice because discrete L-curves often have several small local corners, occasionally associated with clusters of L-curve points. A global point of view of the discrete L-curve is needed in order to find the desired corner of the overall curve.

Figure 1 illustrates this, using the TSVD method on a tiny problem. The top left plot shows a discrete L-curve with 11 points, with the desired global corner at $k = 9$ and with a local corner at $k = 4$ (shown in more detail in the top right plot). For this particular L-curve, the smallest angle between neighboring line segments is attained at $k = 4$; but the L-curve's little "step" here is actually an insignificant part of the overall horizontal part of the curve in this region.

The bottom left plot in Fig. 1 shows a part of the continuous Tikhonov L-curve for the same problem, together with the points of the discrete TSVD L-curve. Clearly the Tikhonov L-curve is not affected very much by the little "step" of the discrete L-curve.

Two algorithms have been proposed for computing the corner of a discrete L-curve, taking into account the need to capture the overall behavior of the curve and avoiding the local corners. The first algorithm, which was described in [8] and implemented in l_corner in [5], fits a 2D spline curve to the points of the discrete L-curve. The curvature of the spline curve is well defined and independent of the parametrization, and the algorithm returns the point on the discrete L-curve closest to the corner of the spline curve.

The spline curve has an undesired tendency to track the unwanted local corners of the discrete L-curve, and therefore a preprocessing stage is added where the L-curve points are first smoothed by means of a local low-degree polynomial. Unfortunately, this smoothing step depends on a few parameters. Hence the overall algorithm is not adaptive, and often it is necessary to hand-tune the parameters of the smoothing process in order to remove the influence of the small local corners, without missing the global corner. If we use the default parameters in [5] then we obtain the spline curve shown in the bottom right plot of Fig. 1 whose corner (indicated by the diamond) is, incorrectly, located at the little "step."

A more recent algorithm, called the *triangle method,* is described in [2]. The key idea here is to consider the following triples of L-curve points:

$$(\mathcal{P}_j, \mathcal{P}_k, \mathcal{P}_p)\,, \qquad j = 1, \ldots, p-2, \quad k = j+1, \ldots, p-1,$$

and identify as the corner the triple where the oriented angle $\theta(j,k,p)$ is minimum. If all angles $\theta(j,k,p)$ are greater than $-\pi/8$ then the L-curve is considered "flat" and the leftmost point is chosen. Note that the leftmost point $\mathcal{P}_p$ of the entire L-curve is always included in the calculations. The authors of the triangle algorithm [2] were not able to provide us with Matlab code, and hence the tests in Section 4 are done using our own Matlab implementation. For the L-curve in Fig. 1 this algorithm returns $k = 8$ which is a good estimate of the optimal $k$.

The main concern with the triangle algorithm is its complexity, which is $\frac{1}{2}(p-1)(p-2)$ and which can be too high when $p$ is large and/or when fast processing is required, e.g., in a real-time application (possibly in connection with updating algorithms). The amount of computation can be reduced by working with a subsampled L-curve, but the subsampling must be done carefully by the user and is not part of the algorithm.

## 3 The Adaptive Pruning Algorithm

An implementation of a robust discrete L-curve criterion should preferably have an average complexity of $O(p \log p)$, and must include a means for adaptively filtering small local corners. The process must be adaptive, because the size or scale of the local phenomena is problem dependent and usually unknown by the user. In addition, the algorithm should not make use of any pre-set parameters.

To achieve the required adaptivity and robustness, our new algorithm consists of two stages. In the first stage we compute corner candidates using L-curves at different scales or resolutions (not knowing a priori which scale is optimal). In the second stage we then compute the overall best corner from the candidates found in the first stage. During the two stages we monitor the results, in order to identify L-curves that lack a corner (e.g., because the problem is well conditioned).

### *3.1 The Overall Algorithm*

The key idea is that if we remove the right amount of points from the discrete L-curve, then the corner can easily be found using the remaining set of points. However, the set of points to be removed is unknown. If too few points are removed we still maintain unwanted local features, and if too many points are removed the corner will be incorrectly located or may disappear.

In the first stage of the algorithm, we therefore work with a sequence of pruned L-curves, that is, curves in which a varying number of points are removed. For each pruned L-curve that is considered convex, we locate two corner candidates. This produces a short list of $r$ candidate points to the corner $\mathcal{P}_{k_1}, \ldots, \mathcal{P}_{k_r}$ and several (or possibly all) of the candidates may be identical. Duplicate entries are removed, and the candidate list is sorted such that the remaining indices satisfy $k_i > k_{i-1}$. Also, to be able to handle $\mathcal{P}_{k_1}$, we augment the list with the first point of the entire L-curve and set $\mathcal{P}_{k_0} = \mathcal{P}_1$.

In the second stage we then pick the best corner from the list of candidates found in the first stage. If the candidate list includes only one point then we are done, otherwise we must choose a single point from the list. We cannot exclude that points on the vertical part of the L-curve are among the candidates, and as a safeguard we therefore seek to avoid any such point. If we traverse the sorted candidate list, then the wanted corner is the last candidate point before reaching the vertical part. If no point lies on the vertical branch of the L-curve, then the last point is chosen. We use two criteria to check for feasible points. A point $\mathcal{P}_{k_i}$, $i = 1, \ldots, r$ is considered lying on the vertical branch of the L-

ADAPTIVE PRUNING ALGORITHM

1. Initialize $\hat{p} = \min(5,\, p-1)$
2. **Stage one:** while $\hat{p} < 2(p-1)$
3. $\quad \hat{p} = \min(\hat{p},\, p-1)$
4. $\quad$ Create a pruned L-curve consisting of the $\hat{p}$ largest line segments.
5. $\quad$ For each corner location routine
6. $\quad\quad$ Locate the corner $\mathcal{P}_k$ using the pruned L-curve.
7. $\quad\quad$ Add the corner to the list: $\mathcal{L} = \mathcal{L} \cup \{\mathcal{P}_k\}$.
8. $\quad \hat{p} = 2\hat{p}$
9. **Stage two:** if $\#\mathcal{L} = 1$ then $k = k_1$; return.
10. Otherwise for $i = 1, \ldots, \#\mathcal{L} - 1$
11. $\quad$ Compute the slope $\phi_i$ associated with point $\mathcal{P}_{k_i}$ in $\mathcal{L}$.
12. If $\max\{\phi_i\} < \pi/4$ then $k = \max\{k_i\}$; return.
13. Otherwise let $k = \min\{k_i : \phi_i > \pi/4 \wedge \theta(k_{i-1},\, k_i,\, k_{i+1}) < 0\}$.

Fig. 2. The overall design of the adaptive pruning algorithm for locating the corner of a discrete L-curve. Here, $\mathcal{P}_k$ denotes a point on the original L-curve, and $\mathcal{P}_{k_i}$ denotes a candidate point in the list $\mathcal{L}$.

curve if the vector $v_{k_{i-1},k_i}$ has a slope greater than $\pi/4$, and the curvature of a candidate point $\mathcal{P}_{k_i}$ is acceptable if the angle $\theta(k_{i-1}, k_i, k_{i+1})$ is negative.

The complete algorithm is shown in Fig. 2. The computation of the corner candidates using each pruned L-curve is done by two separate routines which we describe below, one relying on the angles between subsequent line segments and one aiming at tracking the global vertical and horizontal features of the L-curve. The routine based on angles also checks for correct curvature of the given pruned L-curve, and no corner is returned from this routine if the pruned L-curve is flat or concave. The overall algorithm will always return an index $k$ to a single point which is considered as the corner of the discrete L-curve, unless all the pruned L-curves are found to be concave (in which case the algorithm returns an error message).

### *3.2 Corner Location Based on Angles*

This corner selection strategy was proposed in [9] and is similar in spirit to the guideline of the triangle method [2]. In the pruned L-curve, we find the angle $\theta(k-1, k, k+1)$ which is closest to $-\pi/2$. To explain our method, we consider the angle $\theta_i = \theta(i-1, i, i+1)$ defined in (4), which we can write as

$$\theta_i = s_i|\theta_i|, \quad s_i = \operatorname{sign}(\theta_i), \quad i = 2,\, \ldots,\, p-1$$

If we normalize each vector $v_{j-1,j}$, $j = 2, \ldots, p$ such that $\|v_{j-1,j}\|_2 = 1$ then the two quantities $s_i$ and $|\theta_i|$ are given by

$$s_i = \text{sign}\,(w_i)\,, \quad \text{where} \quad w_i = (v_{i-1,i})_1(v_{i,i+1})_2 - (v_{i-1,i})_2(v_{i,i+1})_1$$

and

$$|\theta_i| = \arccos\, v_{i-1,i}^T v_{i,i+1},$$

which follows from elementary geometry. Here, $(z)_l$ denotes coordinate $l$ of the vector $z$. The corner is then defined by $k = \text{argmin}_i\, |\theta_i + \pi/2|$. Note that $w_i$ carries sufficient information, such that $k = \text{argmin}_i\, |w_i + 1|$. If $\theta_k$ (or equivalently $w_k$) is negative, then the point $\mathcal{P}_k$ is accepted as a corner. Otherwise, the given pruned L-curve is considered flat or concave and no corner is found.

### *3.3 Corner Location Based on Global Behavior*

The approach used here is similar to an idea from [1] in which the corner of the continuous Tikhonov L-curve is defined as the point with smallest Euclidean distance to the "origin" $\mathcal{O}$ of the coordinate system. With $\mathcal{O}$ chosen in a suitable way, it is shown in [1] that the point on the L-curve closest to $\mathcal{O}$ is near the point of maximum curvature.

The main issue is to locate a suitable "origin" in the log-log plot. In [1] it is defined as the point $(\log \|A\, x_{\sigma_n} - b\|_2\,, \log \|x_{\sigma_1}\|_2\,)$ where $x_{\sigma_1}$ and $x_{\sigma_n}$ are the Tikhonov solutions for regularization parameters $\lambda = \sigma_1$ and $\lambda = \sigma_n$, respectively. But given only points on a discrete L-curve, neither the singular values nor their estimates are necessarily known. Instead we seek to identify the "flat" and the "steep" parts of the L-curve, knowing that the corner must lie between these parts.

Define the horizontal vector $v_H = (-1,\, 0)^T$, and let $\hat{p}$ be the number of points on the pruned L-curve. Then we define the slopes $\phi_j$ as the angles between $v_H$ and the normalized vectors $v_{j-1,j}$ for $j = 2, \ldots, \hat{p}$. The most horizontal line segment is identified by $\ell_h = \text{argmin}_j\, |\phi_j|$, and the most vertical one by $\ell_v = \text{argmin}_j |\phi_j + \pi/2|$. Again, $w_j = (v_H)_1(v_{j-1,j})_2 - (v_H)_2(v_{j-1,j})_1 = -(v_{j-1,j})_2$ carries sufficient information and thus $\ell_h = \text{argmin}_j |(v_{j-1,j})_2|$ and $\ell_v = \text{argmin}_j |1 - (v_{j-1,j})_2| = \text{argmax}_j |(v_{j-1,j})_2|$. Hence the horizontal and vertical parts of the curve can be computed very efficiently.

To ensure that the chosen line segments are good candidates for the global behavior of the L-curve, we add the constraint that the horizontal line segment must lie to the right of the vertical one. The "origin" $\mathcal{O}$ is now defined as the intersection between the horizontal line at $\log \|x_{\ell_h}\|_2$ and the line defined by

Table 1
Number of loops and number of line segments per loop in stage one.

| $p$ | $p \le 5$ | $5 < p \le 10$ | $10 < p \le 20$ | $20 < p \le 40$ |
|---|---|---|---|---|
| no. loops $\bar{p}$ | 1 | 2 | 3 | 4 |
| $\hat{p}$ in each loop | $p-1$ | 5, $p-1$ | 5, 10, $p-1$ | 5, 10, 20, $p-1$ |

the vector $v_{\ell_v-1,\ell_v}$. The corner is then selected among all $p$ points on the entire L-curve as the point with the smallest Euclidean distance to this $\mathcal{O}$.

### *3.4 Computational Complexity*

The work in our algorithm is obviously dominated by the two corner-finding algorithms in the loop of stage one. Table 1 shows the number of loops $\bar{p}$ and the number of line segments $\hat{p}$ in each pruned L-curve, and we see that the number of loops is $\bar{p} = \lfloor \log_2(0.4(p-1)) \rfloor \approx \log_2 p$.

The angle-based corner location algorithm (cf. §3.2) involves the computation of the quantities $w_i$ for $i = 1, \ldots, \hat{p}$ in each loop. The total amount of this work is therefore approximately $3(5 + 10 + 20 + \cdots + (p-1)) \approx 3 \cdot 2p$ flops.

The main work in the other corner location algorithm (cf. §3.3) involves, in each loop, the location of the vertical and horizontal branches of the pruned L-curve with $\hat{p}$ line segments, and the corner location via the distance to the origin. The latter involves $5p$ flops. The former involves a double loop with at most $(\hat{p}/2+1)^2 \approx \hat{p}^2/4$ comparisons; but this double loop is terminated as soon as the angle criterion is fulfilled. In the worst case, the work is approximately $5p \cdot \bar{p} + \frac{1}{4}(5^2 + 10^2 + 20^2 + \cdots + (p-1)^2) \approx 5p \log_2 p + \frac{1}{4} \cdot 1.5p^2$ flops.

The total amount of work in the adaptive pruning algorithm is therefore, in the worst case, about $6p + 5p \log_2 p + 0.4p^2$ flops. However, due to the early termination of the double loop mentioned above, we observe experimentally a work load of the order $p \log_2 p$ flops, cf. the next section.

## 4 Numerical Tests and Examples

We illustrate the performance and robustness of our adaptive pruning algorithm, and we compare the algorithm to the two previously described algorithms: l_corner from [5] and the triangle method from [2]. To perform a general comparison of state-of-the-art methods, we also compare with the General Cross Validation (GCV) method, which tries to minimize the predictive mean-

Table 2
The test problems used in our comparison. All problems from REGULARIZATION TOOLS use the default solution, except ilaplace where the third solution is used. All "gallery" matrices are used with the exact solution from the shaw test problem, and prolate is called with parameter 0.05.

| No. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | baart | shaw | wing | hilbert | lotkin | moler | foxgood | gravity | heat | ilaplace | phillips | regutm | prolate |
| Type | Reg. Tools | | | "gallery" | | | Reg. Tools & [4] | | | | | | "gallery" |

square error $\|A\,x_k - b^{\text{exact}}\|_2$, where $b^{\text{exact}}$ is the noise-free right-hand side. In case of TSVD, the parameter $k$ chosen by the GCV method minimizes the function

$$G_k = \frac{\|A\,x_k - b\|_2^2}{(n-k)^2}.$$

While the theory for GCV is well established, the minimum is occasionally very flat resulting in (severely) under-regularized solutions. These problems are described, e.g., in [6, §§7.6–7.7].

### *4.1 Test Problems*

We use eight test problems from REGULARIZATION TOOLS [5] and the problem gravity from [4]. In addition we create four test problems with ill-conditioned matrices from Matlab's "matrix gallery" and the exact solution from the test problem shaw. The test problems are listed in Table 2.

All test problems consist of an ill-conditioned coefficient matrix $A$ and an exact solution $x^{\text{exact}}$ such that the exact right-hand side is given by $b^{\text{exact}} = A\,x^{\text{exact}}$. To simulate measurement errors, the right-hand side $b = b^{\text{exact}} + e$ is contaminated by additive white Gaussian noise $e$ scaled such that the relative noise level $\|e\|_2/\|b^{\text{exact}}\|_2$ is fixed. The TSVD method is used to regularize all test problems. To evaluate the quality of the regularized solutions, we define the best TSVD solution as the solution $x_{k^*}$ where

$$k^* = \text{argmin}_k \|x^{\text{exact}} - x_k\|_2.$$

For problem size $n = 128$ and relative noise level $\|e\|_2/\|b^{\text{exact}}\|_2 = 5 \cdot 10^{-3}$, all test problems are generated with 8 different realizations of the noise. Let $i = 1, \ldots, 13$ denote the problem and $j = 1, \ldots, 8$ the realization number. For each $i$ and $j$, we compute the optimal parameter $k^*_{ij}$ as well as $k^{\text{L}}_{ij}$ from l_corner, $k^{\text{G}}_{ij}$ from the GCV method, $k^{\text{T}}_{ij}$ from the triangle method, and $k^{\text{A}}_{ij}$ from the new adaptive pruning algorithm. The quality of all the solutions is measured by

the quantity

$$Q_{ij}^{\square} = \frac{\|x_{k_{ij}^{\square}} - x^{\text{exact}}\|_2}{\|x_{k_{ij}^{*}} - x^{\text{exact}}\|_2}\,, \qquad \square = \text{A}, \text{L}, \text{G}, \text{T},$$

where A, L, G, and T refer to adaptive pruning algorithm, l_corner, GCV method and triangle method, respectively. The minimum value $Q_{ij}^{\square} = 1$ is optimal, and a value $Q_{ij}^{\square} > 100$ is considered off the scale.

Figure 3 shows the quality measure for all tests. In some occasions, $k_{ij}^{L}$ and $k_{ij}^{G}$ produce regularized solutions that are off the scale; this behavior is well-known because the spline might fit the local behavior of the L-curve and thus find a corner far from the global corner, and the GCV-function can have a very flat minimum. The results from the pruning algorithm are never off the scale, and the results from the triangle method are only off in test problem 9. Overall, both algorithms perform equally good, the triangle method being slightly better than the pruning algorithm for test problem 11. Similar tests were performed for other values of $n$ and different noise levels, the results being virtually the same and therefore not shown here.

It is interesting to observe that GCV behaves somewhat similar for all test problems, whereas the three L-curve algorithms seem to divide the problems into two groups: one that seems easy to treat, and another where the L-curve criterion seems more likely to fail. This effect is more pronounced for small $n$. Figure 4 shows the corner of the L-curve of a realization of test problem nine of size $n = 64$. This L-curve exhibits two corners of which both the pruning algorithm and the triangle algorithm choose the wrong one, leading to large errors. The optimal solution does not lie exactly in the other corner, but slightly to the right of this corner on the horizontal branch of the L-curve. This illustrates that the L-curve heuristic can fail, as mentioned in the Introduction.

Our tests illustrate that the adaptive pruning algorithm is more robust than the l_corner algorithm and the GCV method, and that it performs similar to the triangle method. The tests also illustrate that we cannot always expect to get the optimal regularization parameter by using the L-curve criterion, as this optimum is not always associated with the corner of the L-curve.

As mentioned earlier, the main problem with the triangle method is the complexity of $O(p^2)$, whereas the adaptive pruning algorithm tends to have an average complexity $O(p \log p)$. To illustrate this, all thirteen test problems were run with noise levels $\|e\|_2/\|b^{\text{exact}}\|_2 = 5 \cdot 10^{-2}$, $\|e\|_2/\|b^{\text{exact}}\|_2 = 5 \cdot 10^{-3}$ and $\|e\|_2/\|b^{\text{exact}}\|_2 = 5 \cdot 10^{-4}$ varying the problem size from $n = 16$ to 128, and the number of floating point operations was counted. The results are shown in Fig. 5 for the adaptive pruning algorithm and the triangle method, showing the average over the three noise levels and all test problems. The flop count for the triangle method is about $3p^2$, while it is about $25p \log p$ for the adaptive

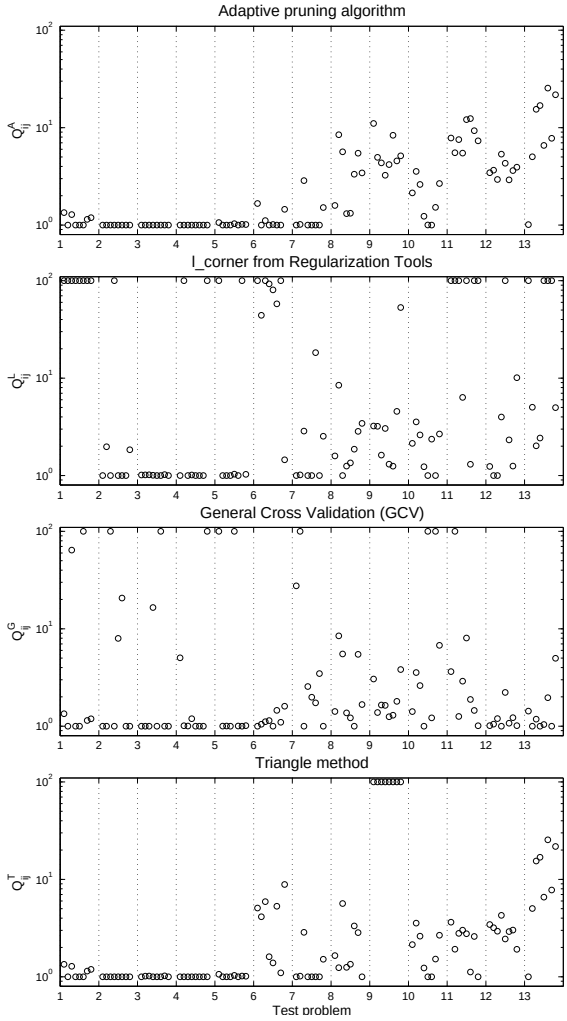



Fig. 3. Quality measure $Q_{ij}^{\square}$ for the four methods and all 13 test problems, with 8 realizations of the noise for a problem size of $n = 128$. A measure of one is optimal, and all values above $10^2$ are set to $10^2$.

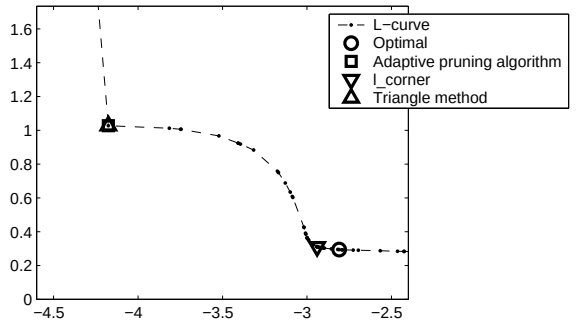



Fig. 4. Problematic L-curve for problem $(i, j) = (5, 9)$. The curve has no simple corner, and the optimal solution lies neither in the corner nor near the corner.

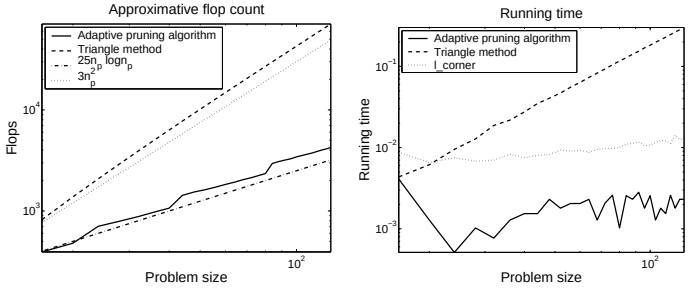



Fig. 5. Left: flop counts for the adaptive pruning algorithm and the triangle method. Right: run times for the three L-curve algorithms.

pruning algorithm.

Figure 5 also shows the average run times. While this measure is sensitive to implementation details, it shows the same trend as the flop counts. Our adaptive pruning algorithm is faster than both the triangle algorithm and l_corner, and for $p = 100$ our algorithm is more than ten times faster than the triangle method.

### *4.2 A Large-Scale Problem*

We also include a large-scale example in the form of an image deblurring problem. The blurring is spatially invariant and separates into column and row blur, and zero boundary conditions are used in the reconstruction. This

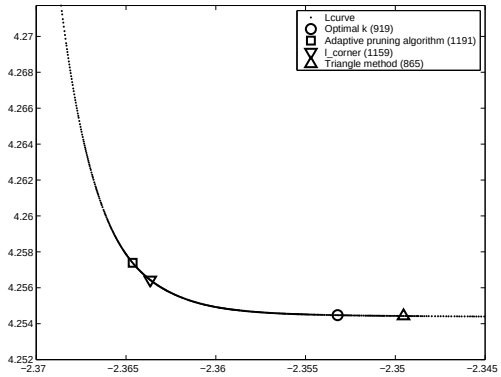



Fig. 6. Corner part of L-curve for large-scale image reconstruction problem. The optimal solution lies on the horizontal part of the curve to the right of the corner, and is denoted by a circle. The number in parenthesis is the corresponding number of CGLS iterations.

leads to a problem where $A$ is a $10^4 \times 10^4$ nonsymmetric Kronecker product of two Toeplitz matrices, and $x$ and $b$ are column-wise stacked images. For the reconstruction we use CGLS with full reorthogonalization.

The L-curve for the image problem after 1500 CGLS iterations is shown in Fig. 6, and we see that both the adaptive pruning algorithm and l_corner find points close to the true corner of the L-curve. The triangle method erroneously identifies a corner far off on the horizontal branch of the L-curve, and its run time is much larger than for the other L-curve algorithms. A simple timing of the methods shows a run time of approximately 28 seconds for the triangle method compared to about half a second for the adaptive pruning algorithm and the l_corner algorithm, using a laptop with a Pentium Centrino 1.4GHz processor. The GCV function is not well-defined for CGLS and therefore is not used here.

## 5 Conclusion

We described a new adaptive algorithm for finding the corner of a discrete L-curve. Numerical examples show that the new algorithm is more robust than the algorithm l_corner from [5] based on spline curve fitting, and faster than the triangle method [2]. It is also shown that the heuristic L-curve algorithm can fail no matter how it is implemented.

**References**

[1] M. Belge, M. E. Kilmer and E. L. Miller, *Efficient determination of multiple regularization parameters in a generalized L-curve framework*, Inverse Problems, 18 (2002), pp. 1161–1183

[2] J. L. Castellanos, S. Gómez and V. Guerra, *The triangle method for finding the corner of the L-curve*, Appl. Num. Math., 43 (2002), pp. 359–373.

[3] M. Hanke, *Limitations of the L-curve method for ill-posed problems*, BIT, 36 (1996), pp. 287–301.

[4] P. C. Hansen, *Deconvolution and regularization with Toeplitz matrices*, Numer. Algo., 29 (2002) pp. 323–378.

[5] P. C. Hansen, *Regularization Tools: A Matlab package for analysis and solution of discrete ill-posed problems*, Numerical Algorithms, 6 (1994), pp. 1–35.

[6] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion,* SIAM, Philadelphia, 1998.

[7] P. C. Hansen, *The L-curve and its use in the numerical treatment of inverse problems;* invited chapter in P. Johnston (Ed.), *Computational Inverse Problems in Electrocardiology*, WIT Press, Southampton, 2001; pp. 119–142.

[8] P. C. Hansen and D. P. O'Leary, *The use of the L-curve in the regularization of discrete ill-posed problems*, SIAM J. Sci. Comput., 14 (1993), pp. 1487–1503.

[9] G. Rodriguez and D. Theis, *An algorithm for estimating the optimal regularization parameter by the L-curve*, Rend. di Matematica, 25 (2005), pp. 69–84.

[10] C. R. Vogel, *Non-convergence of the L-curve regularization parameter selection method*, Inverse Problems, 12 (1996), pp. 535–547.

# SMOOTHING-NORM PRECONDITIONING FOR REGULARIZING MINIMUM-RESIDUAL METHODS

PER CHRISTIAN HANSEN* AND TOKE KOLDBORG JENSEN*

**Abstract.** When GMRES (or a similar minimum-residual algorithm such as RRGMRES, MINRES and MR-II) is applied to a discrete ill-posed problem with a square matrix, in some cases the iterates can be considered as regularized solutions. We show how to precondition these methods in such a way that the iterations take into account a smoothing norm for the solution. This technique is well established for CGLS, but it does not immediately carry over to minimum-residual methods when the smoothing norm is a seminorm or a Sobolev norm. We develop a new technique which works for any smoothing norm of the form $\|L\,x\|_2$ and which preserves symmetry if the coefficient matrix is symmetric. We also discuss the efficient implementation of our preconditioning technique, and we demonstrate its performance with numerical examples in 1D and 2D.

**Key words.** General-form regularization, smoothing norm, regularizing iterations, GMRES, MINRES, weighted pseudoinverse.

**AMS subject classifications.** 65F22, 65F10.

**1. Introduction.** We are concerned with large-scale discrete ill-posed problems with a square coefficient matrix, i.e., ill-conditioned linear systems of the form $A\,x = b$ with $A \in \mathbb{R}^{n\times n}$ and $x,\, b \in \mathbb{R}^n$. These problems typically arise from discretizations of Fredholm integral equations of the first kind, e.g., in computerized tomography, geophysics or image restoration. Due to the ill-conditioning of $A$ and the unavoidable errors in the right-hand side (coming from data), any attempt to compute the "naive" solution $A^{-1}b$ will fail to produce a meaningful solution.

Instead we must use a regularization method to compute a stabilized solution which is less sensitive to the errors. There are many such methods around, and one of the most popular is Tikhonov regularization which amounts to computing

$$x_\lambda = \operatorname{argmin}_x \left\{ \|A\,x - b\|_2^2 + \lambda^2 \,\|L\,x\|_2^2 \right\} = (A^T A + \lambda^2\, L^T L)^{-1} A^T b, \tag{1.1}$$

where the matrix $L$ defines a *smoothing norm* $\|L\,\cdot\|_2$ that acts as a regularizer, and $\lambda$ is the regularization parameter.

For large-scale problems we need iterative methods to compute regularized solutions, and there is a rich literature on CG-based methods for computing the Tikhonov solution via the least-squares formulation of (1.1). More recently we have seen an interest in methods referred to as *regularizing iterations*. These are Krylov subspace methods applied directly to the problem $\min \|A\,x - b\|_2$ or $A\,x = b$ with no additional smoothing norm (such as $\lambda^2 \|L\,x\|_2^2$); instead the projection of the problem onto the Krylov subspace, associated with the method, acts as a regularizer of the solution. See, e.g., [7] and [12] for details.

Probably the newest member of the family of regularizing iteration methods is the GMRES algorithm [15]. If $A$ is symmetric then GMRES is analytically identical to the MINRES algorithm [14], the latter yielding a simpler implementation with a short recursion. Regularizing GMRES and MINRES iterations were recently studied in [1], [2], [3] and [12].

The use of a matrix $L \neq I_n$ in the Tikhonov problem (1.1) can lead to better regularized solutions than the choice $L = I_n$, the explanation being that with a proper

*Informatics and Mathematical Modelling, Building 321, Technical University of Denmark, DK-2800 Lyngby, Denmark (emails: pch@imm.dtu.dk and tkj@imm.dtu.dk).

choice of $L$ the solution $x_\lambda$ is expressed in terms of basis vectors that are better suited for the problem. The choice of $L$ is problem dependent. As demonstrated by Hanke and Hansen [8], the matrix $L$ can be incorporated into the CGLS algorithm for solving $\min \|A\,x - b\|_2$ in such a way that the modified Krylov subspace provides the desired basis for the solution.

The purpose of this paper is to give a rigorous explanation of how we can carry this idea of preconditioning over to regularizing minimum-residual methods for a general smoothing norm $\|L \cdot\|_2$. The hope is that if the minimum-residual methods produce regularized solutions similar to the Tikhonov solutions, then incorporating the smoothing norm will produce solutions comparable to the general-form Tikhonov solutions. The main difficulty is that the smoothing-norm preconditioning from [8] typically involves rectangular matrices and therefore does not immediately carry over to the methods studied here. We shall demonstrate that we can still use the underlying idea, but the practical details and the implementation is different. Our preconditioner has the additional feature that it, when used in connection with symmetric problems, preserves the symmetry of the iteration matrix thus allowing MINRES and MR-II to be used.

Since there is no overall "best" regularization algorithm, we believe that users should preferably have access to a variety of efficient and robust regularization methods. Also, the full understanding of the theoretical properties of regularizing iterations has not emerged, and is a topic of current research. The goal of this paper is therefore not to emphasize preconditioned minimum-residual methods over other regularizing iterations, but instead to demonstrate how it should be defined for a general matrix $L$ and implemented efficiently.

Our paper is organized as follows. Section 2 describes how to incorporate the matrix $L$ into regularizing CGLS iterations via a standard-form transformation based on the $A$-weighted pseudoinverse of $L$. In Section 3 we briefly summarize a method based on augmentation of $L$ to a square matrix. Our main results are given in Section 4 where we introduce our rectangular preconditioning technique, and in Section 5 we demonstrate how to implement the new preconditioner efficiently. Finally, we illustrate our algorithm with 1D and 2D examples in Section 6.

Throughout the paper, $I_q$ is the identity matrix of order $q$, $A^\dagger$ is the pseudoinverse of $A$, $\mathcal{R}(\cdot)$ and $\mathcal{N}(\cdot)$ denote the range and null space of a matrix, and the Krylov subspace is denoted by $\mathcal{K}_k(A, b) = \text{span}\{b, Ab, A^2b, \ldots, A^{k-1}b\}$.

**2. Working with Smoothing Norms.** We first summarize the results from [8] about smoothing norms. The key idea is to transform the general-form Tikhonov problem (1.1) into a problem in standard form

$$\min_x \left\{ \|\bar{A}\,\bar{x} - \bar{b}\|_2^2 + \lambda^2 \|\bar{x}\|_2^2 \right\}.$$

When $L$ is invertible, the standard-form transformation is easy: set $\bar{A} = A\,L^{-1}$ and $\bar{b} = b$, and use $\bar{x} = L\,x \Leftrightarrow x = L^{-1}\bar{x}$.

Often the matrix $L$ is rectangular and therefore not invertible. For example, if the smoothing norm $\|L\,x\|_2$ represents the norm of the first or second derivative of the solution, and if $x$ represents samples of the solution on a regular grid, then as $L$ we use the matrices $L_1 \in \mathbb{R}^{(n-1)\times n}$ and $L_2 \in \mathbb{R}^{(n-2)\times n}$ given by

$$L_1 = \begin{pmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix}, \qquad L_2 = \begin{pmatrix} 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \end{pmatrix}. \tag{2.1}$$

With these rectangular matrices, the smoothing norm $\|L\,x\|_2$ is a seminorm. The matrices $L_1$ and $L_2$ are chosen such that their null spaces

$$\mathcal{N}(L_1) = \text{span}\left\{(1,1,\ldots,1)^T\right\}, \quad \mathcal{N}(L_2) = \text{span}\left\{(1,1,\ldots,1)^T, (1,2,\ldots,n)^T\right\}$$

represent the null spaces of the underlying first and second derivative operators. Obviously, any component of the solution in $\mathcal{N}(L)$ is unaffected by the regularization in (1.1); but since $\mathcal{N}(L_1)$ and $\mathcal{N}(L_2)$ are spanned by very smooth vectors (representing the constant and the linear functions), there is no harm in doing so.

To deal with such rectangular matrices, we assume that the matrix $L \in \mathbb{R}^{p\times n}$ satisfies $\text{rank}(L) = p < n$. Then it is demonstrated in [8] that the standard-form transformation takes the form

$$\bar{A} = A\,L_A^\dagger, \qquad \bar{b} = b - A\,x_0, \qquad x_\lambda = L_A^\dagger \bar{x}_\lambda + x_0,$$

where $L_A^\dagger$ is the $A$-weighted pseudoinverse of $L$, cf. [5], given by

$$L_A^\dagger = E\,L^\dagger, \qquad \text{with} \qquad E = I_n - \big(A\,(I_n - L^\dagger L)\big)^\dagger A, \tag{2.2}$$

and $x_0$ is the component of the solution lying in the null space of $L$,

$$x_0 = \big(A\,(I_n - L^\dagger L)\big)^\dagger\, b = N\,(A\,N)^\dagger\, b,$$

in which $N$ is any matrix of full column rank such that $\mathcal{R}(N) = \mathcal{N}(L)$.

To incorporate the smoothing norm into the framework of regularizing iterations, we apply CGLS to $\|\bar{A}\,\bar{x} - \bar{b}\|_2$, and Hanke and Hansen [8] demonstrated how the CGLS algorithm can be modified in such a way that all operations with $L_A^\dagger$ act as preconditioning. To see this, following §6.1 of [10] we note that if $\mathcal{P}_k$ is the Ritz polynomial associated with $k$ steps of CG applied to $\bar{A}^T\bar{A}\,\bar{x} = \bar{A}^T\bar{b}$, then the iterate $x^{(k)}$ after $k$ steps of the preconditioned CGLS algorithm can be written as

$$x^{(k)} = \mathcal{P}_k\big(L_A^\dagger L_A^{\dagger T} A^T A\big)\, L_A^\dagger L_A^{\dagger T} A^T b + x_0. \tag{2.3}$$

It is now obvious that $L_A^\dagger L_A^{\dagger T}$ acts like a "preconditioner," and efficient methods for implementing this kind of preconditioning for CGLS and other methods are described in [8], [9] and [10, Section 2.3.2]. We refer to the preconditioned version of CGLS as P-CGLS.

In some applications we encounter $L$ matrices with more rows than columns, typically in connection with Sobolev norms such as

$$\|L\,x\|_2^2 = \|L_1\,x\|_2^2 + \|L_2\,x\|_2^2 = \left\| \begin{pmatrix} L_1 \\ L_2 \end{pmatrix} x \right\|_2^2 .$$

In this case an orthogonal factorization of $L$ can often lead to a more efficient implementation. Specifically, if $L = Q\,R$ where $Q$ has orthonormal columns and $R$ is triangular or trapezoidal and has full row rank, then $\|L\,x\|_2 = \|R\,x\|_2$ and we can thus replace $L$ with $R$. This approach can also be used in connection with P-CGLS because $L^\dagger = R^\dagger Q^T$ and $L^\dagger L = R^\dagger R$, and therefore $L_A^\dagger L_A^{\dagger T} = R_A^\dagger R_A^{\dagger T}$, showing that the underlying Krylov subspaces in (2.3) are identical.

Unfortunately, the preconditioner based on $\bar{A}$ cannot be applied to regularizing minimum-residual methods such as MINRES and GMRES because these methods require a square coefficient matrix, which is not the case for $\bar{A}$ when $L$ is noninvertible. Hence we need to develop a different kind of preconditioning for these methods.

**3. The Augmented-Matrix Approach.** To be able to use GMRES/MINRES and the variants RRGMRES [1] and MR-II [6] (i.e., GMRES and MINRES with starting vector $Ab$ instead of $b$), the coefficient matrix must be square. When working with a rectangular $L$, such as in (2.1), it was suggested in [4] to augment it with additional rows to make it square and invertible. For $L_1$ and $L_2$, this approach leads to the augmented $n \times n$ matrices

$$\widehat{L}_1 = \begin{pmatrix} L_1 \\ w^T \end{pmatrix}, \qquad \widehat{L}_2 = \begin{pmatrix} \bar{w}^T \\ L_2 \\ w^T \end{pmatrix}. \tag{3.1}$$

If the additional rows are chosen such that the augmented matrices are invertible, then we can use the matrices $A\,\widehat{L}_1^{-1}$ and $A\,\widehat{L}_2^{-1}$ in connection with the minimum-residual methods. The use of a full-rank matrix $\widehat{L}$ in the standard-form transformation is equivalent to using $\widehat{L}^{-1}$ as a right preconditioner. We refer to [4] for more details about the choices of $w$ and $\bar{w}$.

While this augmented-matrix approach is simple to implement and use, it also has some disadvantages. For example, symmetry of the coefficient matrix $A$ does not carry over to the coefficient matrices $A\widehat{L}_1^{-1}$ and $A\widehat{L}_2^{-1}$, thus excluding the use of MINRES.[1] Moreover, any orthogonal reduction of $L$ changes the iterates because the Krylov subspace changes. For example, if $L$ is nonsingular and $L = QR$ then $\mathcal{K}_k(L^{-1}A, L^{-1}b) = \mathcal{K}_k(R^{-1}Q^TA, R^{-1}Q^Tb) \neq \mathcal{K}_k(R^{-1}A, R^{-1}b)$. This also rules out the use of any $L$ with more rows than columns.

**4. Smoothing-Norm Preconditioning.** As an alternative to the above technique, we now present an approach that works for any rectangular matrix $L \in \mathbb{R}^{p\times n}$ with $p < n$ and which does not require any modifications of the problem. In addition, our approach preserves symmetry, thus allowing short-recurrence implementations such as MINRES and MR-II to be used if $A$ is symmetric.

Our approach is similar in spirit to the technique described in §2 for Tikhonov regularization and CGLS, but the details are different. We refer to the new preconditioned algorithms as SN-X, where X = GMRES, RRGMRES, MINRES or MR-II, and SN is an abbreviation for "smoothing norm."

We start by writing the solution as the sum of the regularized component in $\mathcal{R}(L_A^\dagger)$ and the unregularized component in $\mathcal{N}(L)$,

$$x = L_A^\dagger y + x_0 = L_A^\dagger y + Nz, \tag{4.1}$$

where again $x_0 = N\,(A\,N)^\dagger\,b$, and $N$ is a matrix with full column rank whose columns span $\mathcal{N}(L)$. These columns need not be orthonormal, although this is preferable for numerical computations. The two vectors $y$ and $z = (A\,N)^\dagger b$ are uniquely determined because $L$ and $N$ both have full rank.

Our basic problem $A\,x = b$ can now be formulated as

$$A\,\left(\,L_A^\dagger\,,\,N\,\right)\begin{pmatrix} y \\ z \end{pmatrix} = b.$$

Premultiplication of this system with $\left(\,L_A^\dagger\,,\,N\,\right)^T$ leads to the $2 \times 2$ block system

$$\begin{pmatrix} L_A^{\dagger T} A\,L_A^\dagger & L_A^{\dagger T} A\,N \\ N^T A\,L_A^\dagger & N^T A\,N \end{pmatrix}\begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} L_A^{\dagger T} b \\ N^T b \end{pmatrix}.$$

[1] For symmetric $A$, one might instead consider applying MINRES to the system $\widehat{L}^{-T}A\widehat{L}^{-1}x = \widehat{L}^{-T}b$, where $\widehat{L}^{-T}A\widehat{L}^{-1}$ is symmetric.

We eliminate $z$ from this system by forming the Schur complement system $S\,y = d$ with $S$ and $d$ given by

$$S = L_A^{\dagger T} A\, L_A^{\dagger} - L_A^{\dagger T} A\, N (N^T A\, N)^{-1} N^T A\, L_A^{\dagger} = L_A^{\dagger T} P\, A\, L_A^{\dagger}, \tag{4.2}$$

$$d = L_A^{\dagger T} b - L_A^{\dagger T} A\, N (N^T A\, N)^{-1} N^T b = L_A^{\dagger T} P\, b, \tag{4.3}$$

where we have introduced $P = I_n - AN(N^T AN)^{-1} N^T$. We shall now study the Schur system $S\,y = d$ in more detail.

THEOREM 4.1. *If $\mathcal{R}(L^T)$ and $\mathcal{R}(AN)$ are complementary subspaces*[2] *then*

$$P = I_n - AN(N^T AN)^{-1} N^T \tag{4.4}$$

*is the oblique projector onto $\mathcal{R}(L^T)$ along $\mathcal{R}(AN)$.*

*Proof.* The matrix $I_n - P$ is idempotent because

$$(I_n - P)^2 = AN(N^T AN)^{-1} N^T AN(N^T AN)^{-1} N^T = I_n - P,$$

and hence it is a projector. Since $I_n - P$ is nonsymmetric, it is an oblique projector, and it is easy to see that the projection is onto $\mathcal{R}(AN)$ with $\mathcal{R}(L^T)$ contained in the null space. The assumption that $\mathcal{R}(L^T)$ and $\mathcal{R}(AN)$ are complementary subspaces ensures that $P$ is an oblique projector onto $\mathcal{R}(L^T)$ along $\mathcal{R}(AN)$. ∎

Smoothing-norm preconditioning for GMRES amounts to applying GMRES to the Schur complement system $Sy = d$. We emphasize that, similarly with CGLS, the purpose of this preconditioning is to provide a more desirable Krylov subspace for the regularized solution.

When we apply GMRES to the Schur system $Sy = d$ then there exists a polynomial $\widetilde{\mathcal{P}}_k$ such that the solution after $k$ iterations is given by

$$y^{(k)} = \widetilde{\mathcal{P}}_k\Big(L_A^{\dagger T} PA\, L_A^{\dagger}\Big)\, L_A^{\dagger T} P\, b.$$

The corresponding vector $x^{(k)}$ is given by $x^{(k)} = L_A^{\dagger} y^{(k)} + x_0$, and we therefore obtain the SN-GMRES iterate

$$\begin{aligned} x^{(k)} &= L_A^{\dagger} \widetilde{\mathcal{P}}_k\Big(L_A^{\dagger T} PAL_A^{\dagger}\Big)\, L_A^{\dagger T} P\, b + x_0 \\ &= \widetilde{\mathcal{P}}_k\Big(L_A^{\dagger} L_A^{\dagger T} PA\Big)\, L_A^{\dagger} L_A^{\dagger T} P\, b + x_0, \end{aligned} \tag{4.5}$$

showing that $x^{(k)} - x_0$ lies in the Krylov subspace $\mathcal{K}_k(L_A^{\dagger} L_A^{\dagger T} PA, L_A^{\dagger} L_A^{\dagger T} Pb)$. We note that the iterates of RRGMRES take the same form as for GMRES, except the polynomial coefficients are different, and thus RRGMRES can also be used on the Schur system.

Although the polynomial expressions for the preconditioned CGLS and GMRES methods in (2.3) and (4.5) are similar in essence, the solutions obtained from the two methods are different, due to CGLS being a Ritz-Galerkin method and GMRES being a minimum-residual method. Even when $L$ is invertible, the two approaches produce different iterates. Furthermore, the oblique projector $P$ also indicates that the SN-X algorithms do not solve the same problem as P-CGLS. Nevertheless, as we illustrate

[2]The subspaces $\mathcal{R}(L^T) \subseteq \mathbb{R}^n$ and $\mathcal{R}(AN) \subseteq \mathbb{R}^n$ are complementary if $\mathcal{R}(L^T) + \mathcal{R}(AN) = \mathbb{R}^n$ and $\mathcal{R}(L^T) \cap \mathcal{R}(AN) = \{0\}$, see, e.g., [13, Sec. 5.9].

in §6, our preconditioned algorithms are able to produce good solutions for certain problems.

We emphasize that the two main difficulties with the augmented-matrix approach from §3 are both satisfactorily dealt with in this new approach. For a symmetric matrix $A$, the matrix $L_A^{\dagger T} P\, A\, L_A^{\dagger}$ is also symmetric, which follows from the symmetry of $PA = A - AN\,(N^T A\, N)^{-1} N^T A$. This symmetry allows us to use MINRES or MR-II on the Schur system, resulting in SN-MINRES and SN-MR-II. The new approach also allows us to use any rectangular $L$, including those with more rows than columns via the orthogonal reduction $L = QR$ mentioned in §2.

**5. Implementation Issues.** In this section we consider some issues that are important for the efficient implementation of the SN-X algorithms. We start with a theorem that simplifies the Schur system.

THEOREM 5.1. *If the requirements in Theorem 4.1 are satisfied, then the Schur system $S\,y = d$ given by (4.2)–(4.3) takes the simpler form*

$$L^{\dagger T} P\, A\, L^{\dagger} y = L^{\dagger T} P\, b, \tag{5.1}$$

*with $P$ given by (4.4).*

*Proof.* ¿From the relation $\big(A(I_n - L^{\dagger}L)\big)^{\dagger} = (AN N^{\dagger})^{\dagger} = N(AN)^{\dagger}$ it follows that the matrix $E$ in (2.2) can be written as $E = I_n - N(AN)^{\dagger}A$. Moreover,

$$\begin{aligned} A^T (AN)^{\dagger T} N^T P &= A^T (AN)^{\dagger T} N^T - A^T (AN)^{\dagger T} N^T AN (N^T AN)^{-1} N^T \\ &= A^T (AN)^{\dagger T} N^T - A^T (AN)^{\dagger T} N^T = 0 \end{aligned}$$

and therefore $E^T P = \big(I_n - A^T (AN)^{\dagger T} N^T\big) P = P$. We also have the relation

$$\begin{aligned} P\, A\, N (AN)^{\dagger} A &= \big(I_n - AN(N^T AN)^{-1} N^T\big) AN (AN)^{\dagger} A \\ &= AN(AN)^{\dagger} A - AN(AN)^{\dagger} A = 0 \end{aligned}$$

and thus $PAE = PA\big(I_n - N(AN)^{\dagger}A\big) = PA$. Inserting these relations and $L_A^{\dagger} = EL^{\dagger}$ into the Schur system, we obtain (5.1). □

This theorem has an important impact on the numerical implementation of our preconditioner, because the weighted pseudoinverse $L_A^{\dagger}$ can be replaced by the ordinary pseudoinverse $L^{\dagger}$ in the computations. The weighted pseudoinverse $L_A^{\dagger}$ is needed only in the back-transformation (4.1).

Turning to the details of the implementation, we need to compute $x_0$ efficiently. This is done via the QR factorization of the matrix $AN$ which is always "skinny" for the low-dimensional null spaces associated with the derivative operators:

$$\begin{aligned} &1.\quad AN = Q_0 R_0 \text{ (skinny QR factorization)} \\ &2.\quad x_0 \leftarrow N\, R_0^{-1} Q_0^T b. \end{aligned} \tag{5.2}$$

We also need an efficient technique for multiplications with $P$ from (4.4), which basically amounts to a number of "skinny" matrix-vector products. Using again the QR factorization of $AN$ we obtain

$$AN\,(N^T AN)^{-1} N^T = Q_0 R_0 (N^T Q_0 R_0)^{-1} N^T = Q_0\,(N^T Q_0)^{-1} N^T$$

and thus the product $P\,x$ is computed as

$$P\,x = x - Q_0\,(N^T Q_0)^{-1} N^T x,$$

where a pre-computed factorization of the small square matrix $N^T Q_0$ should be used. Assuming that $N$ has orthonormal columns, the smallest singular value of $N^T Q_0$ is equal to cosine of the subspace angle between $\mathcal{N}(L)$ and $\mathcal{R}(AN)$. Our experience is that this angle is usually small (because the smooth basis vectors of the two subspaces resemble each other) and consequently $N^T Q_0$ is well conditioned – but there is no guarantee that this is always the case.

The complete algorithm for performing the multiplication $v = L^{\dagger T} P A L^{\dagger} y$ in the SN-X algorithms thus takes the form

$$\begin{aligned} &1. \quad v_1 \leftarrow A\,(L^{\dagger} y) \\ &2. \quad v_2 \leftarrow Q_0\,(N^T Q_0)^{-1} N^T v_1 \\ &3. \quad v \leftarrow L^{\dagger T}(v_1 - v_2). \end{aligned} \tag{5.3}$$

The cost of working with the Schur complement system is therefore for each iteration: one multiplication with $A$, one with $L^{\dagger}$, one with $L^{\dagger T}$, and one with the oblique projector $P$. The preconditioning technique is feasible when the computation of $L^{\dagger} y$ and $L^{\dagger T}(v_1 - v_2)$ can be implemented efficiently, and the null space $\mathcal{N}(L)$ has low dimension such that multiplication with $P$ is inexpensive. The remaining work, i.e., reorthogonalization and update of the residual norm and the solution, etc., is identical to applying the minimum-residual method to a non-preconditioned system. A complete SN-X algorithm thus takes the form

$$\begin{aligned} &1. \quad \text{Use (5.2) to compute } Q_0,\ R_0 \text{ and } x_0. \\ &2. \quad \text{Run } k \text{ steps of alg. X on (5.1), using (5.3), to compute } y^{(k)}. \\ &3. \quad \text{Set } x^{(k)} = L_A^{\dagger} y^{(k)} + x_0. \end{aligned} \tag{5.4}$$

While not fully documented in [8]–[10], P-CGLS – in addition to the multiplications with $A$ and $A^T$ – also requires multiplications with $L^{\dagger}$ and its transpose, as well as one multiplication with the matrix $E$ (which is also an oblique projector). Hence the overall work for preconditioning the SN-X algorithms is essentially the same as that for P-CGLS.

We use an example from 2D problems to illustrate how to work with a rank-deficient $L$ with more rows than columns. Assume that $X \in \mathbb{R}^{M \times N}$ is the 2D solution, and that we use the Sobolev norm $\|L_{d_2} X\|_{\mathrm{F}}^2 + \|X\, L_{d_1}^T\|_{\mathrm{F}}^2$, where $L_{d_1} \in \mathbb{R}^{(N-d_1)\times N}$ and $L_{d_2} \in \mathbb{R}^{(M-d_2)\times M}$ are the matrices in (2.1). Then $L$ takes the form

$$L = \begin{pmatrix} L_{d_1} \otimes I_M \\ I_N \otimes L_{d_2} \end{pmatrix}, \tag{5.5}$$

and we note that $L$ has more rows than columns and is rank deficient. The following theorem shows how to proceed via the SVDs of the "small" matrices $L_{d_1}$ and $L_{d_2}$.

THEOREM 5.2. *Let $L$ be given by (5.5), and let $L_{d_1} = U_{d_1}\,\Sigma_{d_1}\,V_{d_1}^T$ and $L_{d_2} = U_{d_2}\,\Sigma_{d_2}\,V_{d_2}^T$ be the SVDs of $L_{d_1}$ and $L_{d_2}$, respectively. Then $\|L\,x\|_2 = \|L_D\,x\|_2$ with*

$$L_D = D\,(V_{d_1} \otimes V_{d_2})^T, \tag{5.6}$$

*where $D \in \mathbb{R}^{MN \times MN}$ is a nonnegative diagonal matrix satisfying*

$$D^2 = \Sigma_{d_1}^T \Sigma_{d_1} \otimes I_M + I_N \otimes \Sigma_{d_2}^T \Sigma_{d_2}. \tag{5.7}$$

*Furthermore, define the matrix splitting* $V_{d_i} = \left( \overline{V}_{d_i} \, , \, N_{d_i} \right)$ *such that* $\mathcal{N}(L_{d_i}) = \mathcal{R}(N_{d_i})$ *for* $i = 1, 2$. *Then a basis for* $\mathcal{N}(L) = \mathcal{N}(L_D)$ *is given by the columns of*

$$N = N_{d_1} \otimes N_{d_2}. \tag{5.8}$$

*Proof.* Inserting the SVDs of $L_{d_1}$ and $L_{d_2}$ and using $I_N = V_{d_1} V_{d_1}^T$ and $I_M = V_{d_2} V_{d_2}^T$ we obtain

$$\begin{aligned} L &= \begin{pmatrix} L_{d_1} \otimes I_M \\ I_N \otimes L_{d_2} \end{pmatrix} = \begin{pmatrix} (U_{d_1} \, \Sigma_{d_1} \, V_{d_1}^T) \otimes (V_{d_2} \, V_{d_2}^T) \\ (V_{d_1} \, V_{d_1}^T) \otimes (U_{d_2} \, \Sigma_{d_2} \, V_{d_2}^T) \end{pmatrix} \\ &= \begin{pmatrix} U_{d_1} \otimes V_{d_2} & 0 \\ 0 & V_{d_1} \otimes U_{d_2} \end{pmatrix} \begin{pmatrix} \Sigma_{d_1} \otimes I_M \\ I_N \otimes \Sigma_{d_2} \end{pmatrix} (V_{d_1} \otimes V_{d_2})^T. \end{aligned}$$

Since the middle matrix consists of two "stacked" diagonal matrices, we can easily determine an orthogonal matrix $Q_D$ (consisting of Givens rotations) and a diagonal matrix $D$ such that

$$Q_D^T \begin{pmatrix} \Sigma_{d_1} \otimes I_M \\ I_N \otimes \Sigma_{d_2} \end{pmatrix} = \begin{pmatrix} D \\ 0 \end{pmatrix}$$

and it is no restriction to assume the diagonal elements of $D$ are nonnegative. Hence

$$L = \begin{pmatrix} U_{d_1} \otimes V_{d_2} & 0 \\ 0 & V_{d_1} \otimes U_{d_2} \end{pmatrix} Q_D \begin{pmatrix} D \\ 0 \end{pmatrix} (V_{d_1} \otimes V_{d_2})^T$$

and we obtain $\|L \, x\|_2 = \|L_D \, x\|_2$. Eq. (5.7) follows from the relation

$$D^2 = D^T D = \begin{pmatrix} D \\ 0 \end{pmatrix}^T \begin{pmatrix} D \\ 0 \end{pmatrix} = \begin{pmatrix} \Sigma_{d_1} \otimes I_M \\ I_N \otimes \Sigma_{d_2} \end{pmatrix}^T \begin{pmatrix} \Sigma_{d_1} \otimes I_M \\ I_N \otimes \Sigma_{d_2} \end{pmatrix}.$$

Regarding the null space $\mathcal{N}(L) = \mathcal{N}(L_D)$, it is easily seen from (5.6) that the null space vectors are given by the columns of $V_{d_1} \otimes V_{d_2}$ for which the diagonal elements of $D$ are zero. This leads directly to (5.8). □

The consequences of this theorem are that we can substitute the structured matrix $L_D$ for $L$ in the Tikhonov problem (1.1), and that we have a simple basis for $\mathcal{N}(L)$. The approach can be used in both the P-CGLS and the SN-X algorithms and leads to increased efficiency, while $L$ matrices of this form are *not* applicable in the augmented-matrix approach.

**6. Numerical Experiments.** We include two test problems to illustrate the performance of the new preconditioning technique for regularizing minimum-residual methods. The first example is a synthetic two-dimensional problem with a symmetric coefficient matrix and an $L$ of the form (5.5); the second is a finite-element model of a steady-state heat distribution problem with a nonsymmetric coefficient matrix and a periodic solution. In both examples we calculate the relative error of the regularized solutions $x^{(k)}$ compared to the exact solution $x$, i.e.,

$$\epsilon^{(k)} = \|x^{(k)} - x\|_2 \, / \, \|x\|_2, \tag{6.1}$$

where $x^{(k)}$ is the $k$th iterate. The best regularized solution is always defined as the solution for which $\epsilon^{(k)}$ is smallest.

**6.1. Two-Dimensional Example.** The two-dimensional example is a synthetic problem generated from the deriv2 test problem in REGULARIZATION TOOLS [9]. The coefficient matrix $A \in \mathbb{R}^{37500\times 37500}$ is generated as the Kronecker product of $A_1 \in \mathbb{R}^{250\times 250}$ and $A_2 \in \mathbb{R}^{150\times 150}$, each being a discretization of Green's function for the second derivative. The resulting coefficient matrix is symmetric, and thus MINRES and MR-II can be used.

The exact solution $X$ has size $M \times N = 150 \times 250$, and it consists of the sum of a linear function in one direction and a quadratic function in the other. The MATLAB code for generating the exact solution is:

```
s = linspace(-1,1,250); t = linspace(-1,1,150);
[s,t] = meshgrid(s,t);
X = s + t.^2;
```

and then $x$ is obtained by stacking the columns of $X$. The right-hand side is computed by $b = Ax+e$, in which $e$ is white Gaussian noise scaled such that $\|e\|_2/\|Ax\|_2 = 10^{-2}$. Using the Kronecker products, all computations can be carried out without explicitly forming the large matrix $A$, instead using $A_1$ and $A_2$.

The top plots in Fig. 6.1 show the exact solution and the right-hand side as two-dimensional images. Figure 6.1 also shows the best regularized solutions using MINRES, MR-II, CGLS, SN-MINRES, SN-MR-II, and P-CGLS as measured by (6.1). For the preconditioned algorithms we use the matrix $L$ in (5.5) with $d_1 = d_2 = 1$, corresponding to first derivative smoothing in both directions, and its null space is spanned by the constant image.

The six algorithms produce solutions with different properties; all the preconditioned algorithms give much better regularized solutions than the non-preconditioned algorithms, and MINRES gives by far the most noisy solution. All the non-preconditioned solutions tend to go to zero near the edges of the domain; this behavior comes from the fact that the basis for these solutions are the Krylov vectors which all tend to zero at the edges. Using the matrix $L$ from (5.5) as preconditioner, the Krylov subspaces are changed to favor the particular properties of this problem, resulting in better reconstructions. The best SN-MR-II solution is similar to the best solution obtained by P-CGLS, both regarding the quality and the number of iterations. It is worth to note that even though the cost of applying the preconditioner is about the same for P-CGLS and SN-MR-II, P-CGLS needs an additional matrix-vector multiplication with $A^T$ in each iteration, which makes SN-MR-II somewhat faster.

**6.2. Steady-State Heat Distribution.** This test problem from [11] involves a partial differential equation that describes the steady-state heat distribution in a two-dimensional domain $\Omega$ with inner and outer boundaries $\partial\Omega_\mathrm{i}$ and $\partial\Omega_\mathrm{o}$. The forward problem takes the form

$$\nabla^2 f(z) = 0 \quad , \quad z \in \Omega \qquad\qquad \begin{cases} f(z) = f_\mathrm{i}(z) & , \quad z \in \partial\Omega_\mathrm{i} \\ \frac{\partial}{\partial n} f(z) = 0 & , \quad z \in \partial\Omega_\mathrm{o} \end{cases}$$

where $f_\mathrm{i}$ is the temperature on the inner boundary, and $\frac{\partial}{\partial n}$ denotes the normal derivative on the outer boundary. The inverse problem is to determine the temperature distribution on the inner boundary from measurements of the temperature on the outer boundary.

In this example, the inner boundary $\partial\Omega_\mathrm{i}$ is an ellipse with semiaxes of length 1.2 and 0.8, while the outer boundary $\partial\Omega_\mathrm{o}$ is a square with side length 4, see Fig. 6.2. We

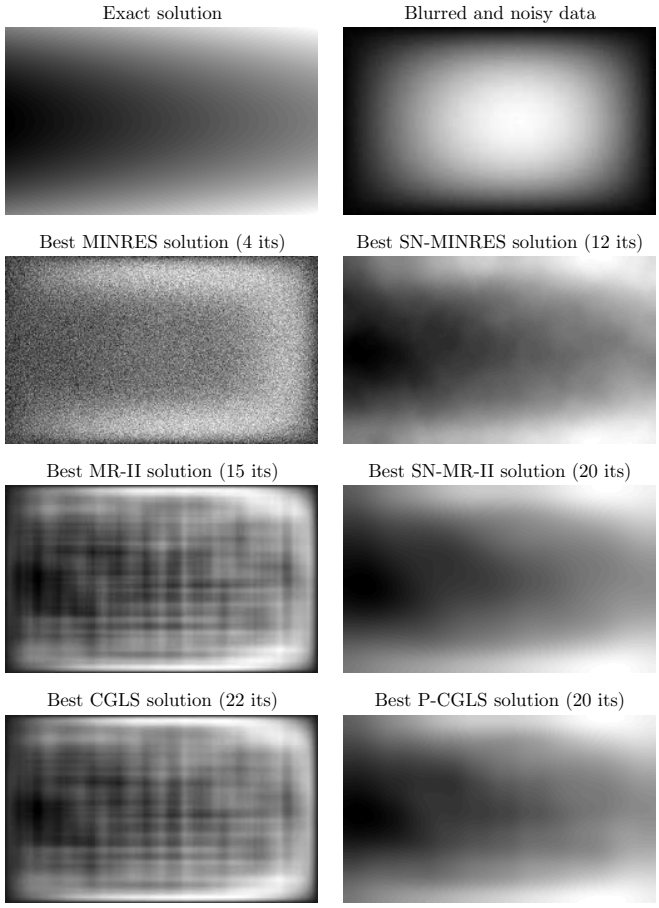



FIG. 6.1. *Top row: exact solution and righ-hand side for the 2D test problem with a symmetric coefficient matrix. The remaining rows show the best regularized solutions using standard and preconditioned algorithms. The preconditioning uses the matrix $L$ in (5.5) with $d_1 = d_2 = 1$ corresponding to first derivative smoothing in both the vertical and horizontal directions.*

use a matrix-free implementation in which $A$, the forward computation, is a finite-element model that maps the temperature $f_{\mathrm{i}}(z)$ on the inner boundary $\partial\Omega_{\mathrm{i}}$ to the temperature $f_{\mathrm{o}}(z)$ on the outer boundary $\partial\Omega_{\mathrm{o}}$.

The exact solution vector $x$ consists of values of the temperature on the inner

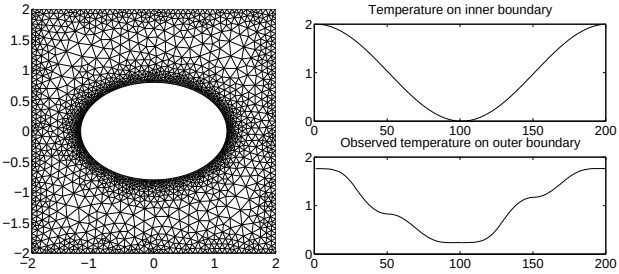



FIG. 6.2. *Test problem with steady-state heat distribution. Left: the geometry and the finite-element mesh. Right: exact temperature $f_i$ on the inner boundary (top), and measured noisy temperature $f_o$ on the outer boundary (bottom).*

boundary in the $n = 200$ grid points. The right-hand side, consisting of the temperature in the $n$ grid points of the outer boundary, is then computed as $b = Ax+e$, where $e$ is white Gaussian noise scaled such that $\|e\|_2/\|Ax\|_2 = 10^{-3}$. Both temperature profiles are also shown in Fig. 6.2.

Using $n$ points on both $\partial\Omega_i$ and $\partial\Omega_o$, the operation with the square matrix $A$ involves solving the forward problem via the finite-element model. A similar discrete model for operation with the transposed matrix is not simple to derive, and therefore P-CGLS is not directly applicable. On the other hand, GMRES and RRGMRES are natural methods of choice for solving the inverse problem.

The two top rows in Fig. 6.3 show the optimal solutions obtained with GMRES and RRGMRES using $L = I_n$ and $L = L_2$. The third row shows the similar results with $L$ equal to the augmented matrix $\widehat{L}_2$ from (3.1) with $\bar{w} = \alpha(-2, 1, 0, \ldots, 0)^T$, $w = \alpha(0, \ldots, 0, 1, -2)^T$ and $\alpha = 4.29 \cdot 10^{-4}$. The $\alpha$-parameter can be considered as a special regularization parameter for the added rows and must be chosen appropriately in addition to the number of iterations. Selecting the optimal $\alpha$ is in general not an easy problem. For this constructed test problem we compute the optimal solutions for a range of $\alpha$ values, and the chosen $\alpha$ is the one that gives rise to the overall best regularized solution. Figure 6.4 shows how the optimal GMRES solutions vary with $\alpha$ – the situation is very similar for RRGMRES.

For all three choices of $L$, RRGMRES performs better than GMRES. Using a smoothing norm with $L = L_2$ or $L = \widehat{L}_2$ improves the solutions by about an order of magnitude. With these two choices of $L$, the augmented-matrix approach seems to performs better than the SN-approach. On the other hand, Fig. 6.4 shows that a small change in $\alpha$ will result in worse solutions. An obvious choice of $\alpha$ would be to choose a value so small that the influence of the added rows is negligible while still keeping $\widehat{L}_2$ invertible, e.g., $\alpha = 10^{-8}$. This corresponds to the flat part of the plot in Fig. 6.4, and an optimal solution with a higher relative error. The dependence on $\alpha$ illustrates the importance of choosing wise extensions to the $L$-matrices. Especially, the extension proposed in [4, Eq. (13)] ($\widehat{L}_2$ with $\alpha = 1$) is seen to perform badly for this particular problem.

However, the quality of the solution can be improved further by taking into account the periodicity of the solution. A careful study shows that the errors in all

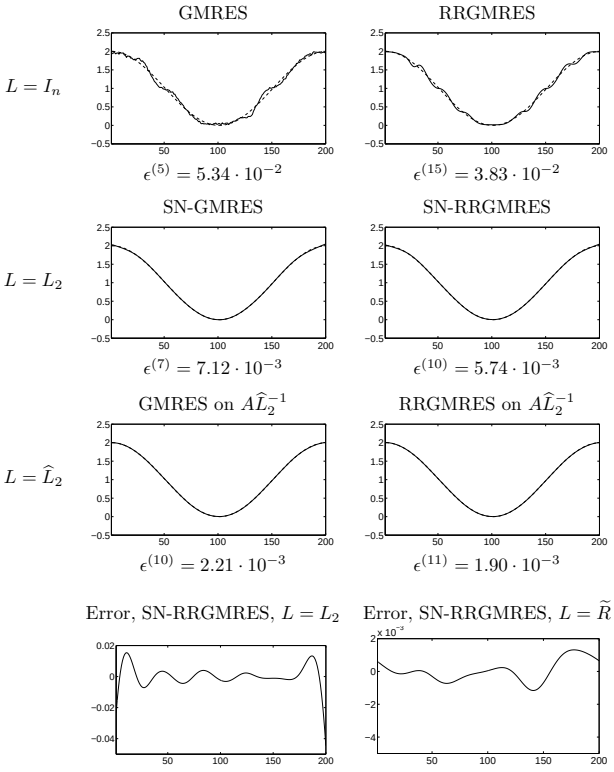



FIG. 6.3. *Three top rows: solutions to the steady-state heat distribution problem computed with standard and preconditioned GMRES and RRGMRES, and with different smoothing norms. The numbers $\epsilon^{(k)}$ below the plots are the optimal relative errors after $k$ iterations. Bottom row: the error $x - x^k$ in the SN-RRGMRES solution for two choices of $L$.*

the solutions grow towards the edges; this is illustrated by the bottom left plot in Fig. 6.3 which shows the error in the SN-RRGMRES solution with $L = L_2$. A natural requirement is therefore to add the constraint that the second derivative should also be minimized across the edges of the periodic solution. Hence, we use a circulant

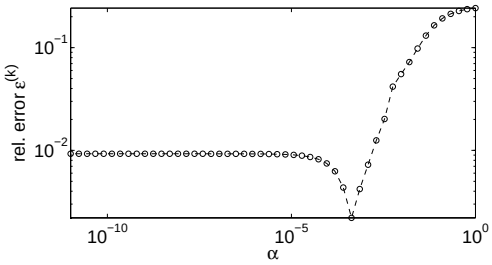



FIG. 6.4. *Best obtainable relative errors $\epsilon^{(k)}$ of GMRES solutions using $\widehat{L}_2$ with $\bar{w} = \alpha(-2,\, 1,\, 0,\, \ldots,\, 0)^T$, and $w = \alpha(0,\, \ldots,\, 0,\, 1,\, -2)^T$ for a range of values of $\alpha$.*

version of $L_2$ given by

$$\widetilde{L}_2 = \begin{pmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 1 & & & 1 & -2 \end{pmatrix} \in \mathbb{R}^{n \times n},$$

which corresponds to $L_2$ with periodic boundary conditions. This matrix is singular with $\operatorname{rank}(\widetilde{L}_2) = n - 1$ and its null space is given by

$$\mathcal{N}(\widetilde{L}_2) = \operatorname{span}\left\{(1,\, 1,\, \ldots,\, 1)^T\right\}.$$

Since $\widetilde{L}_2$ is singular, it cannot be used in the augmented-matrix approach. In order to use the SN-approach, we compute the skinny QR-factorization $\widetilde{L}_2 = \widetilde{Q}\widetilde{R}$ with $\widetilde{R}$ trapezoidal, and use $L = \widetilde{R}$.

Using this choice of $L$, the smallest error $\epsilon^{(6)} = 6.28 \cdot 10^{-4}$ in the SN-GMRES solution is obtained after 6 iterations, while the smallest error $\epsilon^{(5)} = 5.21 \cdot 10^{-4}$ in the SN-RRGMRES solution is obtained after 5 iterations. In both cases, the relative error is reduced by an order of magnitude, compared to using $L_2$, and is also superior compared to the approach using the augmented $\widehat{L}_2$ with $\alpha = 4.29 \cdot 10^{-4}$. Furthermore, the number of iterations is reduced. The bottom right plot in Fig. 6.3, which shows the error in the SN-RRGMRES solution with $L = R$, illustrates the reduction of the error, primarily because the errors near the edges are reduced significantly.

**7. Conclusion.** We presented a new preconditioning technique for regularizing minimum-residual methods such that it corresponds to using a smoothing norm $\|Lx\|_2$ in the Tikhonov formulation, and the matrix $L$ is allowed to be both rank deficient and rectangular. Our algorithm preserves symmetry when the coefficient matrix is symmetric, thus allowing the use of MINRES and MR-II where appropriate. Our algorithm is computationally feasible when the dimension of $\mathcal{N}(L)$ is small and computations with $L^\dagger$ can be implemented efficiently. We showed an example of an $L$ matrix for two-dimensional problems where this is achieved. We also demonstrated how to implement the algorithm efficiently, and we gave numerical examples in 1D and 2D that illustrate the use and performance of the new preconditioner.

Our numerical examples illustrate that the proposed SN-approach can provide an improvement in the solution's accuracy compared to standard minimum-residual methods, that the SN-approach works for a larger class of smoothing norms than the augmented-matrix approach, and that preconditioned minimum-residual methods can be computationally attractive alternatives to preconditioned CGLS (e.g., when $A^T$ is not directly available).

**Acknowledgements.** We would like to thank the referees for comments that considerably improved the presentation and the numerical examples.

REFERENCES

[1] D. Calvetti, B. Lewis, and L. Reichel, *GMRES-type methods for inconsistent systems*, Elsevier, Lin. Alg. and its App., 316 (2000), pp. 157–169.
[2] ———, *GMRES, L-curves, and discrete ill-posed problems*, BIT, 42 (2002), pp. 44–65.
[3] ———, *On the regularizing properties of the GMRES method*, Numer. Math., 91 (2002), pp. 605–625.
[4] D. Calvetti, L. Reichel, and A. Shuibi, *Invertible smoothing preconditioners for linear discrete ill-posed problems*, App. Num. Math., 54 (2005), pp. 135–149.
[5] L. Eldén, *A weighted pseudoinverse, generalized singular values, and constrained least squares problems*, BIT, 22 (1982), pp. 487–501.
[6] M. Hanke, *Conjugate Gradient Type Methods for Ill-Posed Problems*, Longman, Harlow, UK, 1995.
[7] ———, *On Lanczos based methods for the regularization of discrete ill-posed problems*, BIT, 41 (2001), pp. 1008–1018.
[8] M. Hanke and P. C. Hansen, *Regularization methods for large-scale problems*, Surveys Math. Industry, 3 (1993), pp. 253–315.
[9] P. C. Hansen, *Regularization Tools: A Matlab package for analysis and solution of discrete ill-posed problems*, Numer. Algo., 6 (1994), pp. 1–35.
[10] ———, *Rank-Deficient and Discrete Ill-Posed Problems*, SIAM, Philadelphia, 1998.
[11] M. Jacobsen, *Modular regularization algorithms*. PhD thesis, Informatics and Mathematical Modelling, Technical University of Denmark, 2004.
[12] M. E. Kilmer and G. W. Stewart, *Iterative regularization and MINRES*, SIAM J. Matrix Anal. Appl., 21 (1999), pp. 613–628.
[13] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM, Philadelphia, 2000.
[14] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal, 12 (1975), pp. 43–71.
[15] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

BIT Numerical Mathematics (2006)46:000-000
DOI:10.1007/s10543-000-0000-x


# ITERATIVE REGULARIZATION WITH MINIMUM-RESIDUAL METHODS*

T. K. JENSEN and P. C. HANSEN

*Informatics and Mathematical Modelling, Technical University of Denmark, Building 321, DK-2800 Lyngby, Denmark.*
*emails: toke.jensen@gmail.dk, pch@imm.dtu.dk*

**Abstract.**

We study the regularization properties of iterative minimum-residual methods applied to discrete ill-posed problems. In these methods, the projection onto the underlying Krylov subspace acts as a regularizer, and the emphasis of this work is on the role played by the basis vectors of these Krylov subspaces. We provide a combination of theory and numerical examples, and our analysis confirms the experience that MINRES and MR-II can work as general regularization methods. We also demonstrate theoretically and experimentally that the same is not true, in general, for GMRES and RRGMRES – their success as regularization methods is highly problem dependent.

*AMS subject classification (2000):* 65F22, 65F10.

*Key words:* Iterative regularization, discrete ill-posed problems, GMRES, RRGMRES, MINRES, MR-II, Krylov subspaces.

## 1 Introduction

We study iterative methods for solution of large-scale discrete ill-posed problems of the form $Ax = b$ with $A \in \mathbb{R}^{n\times n}$ arising from discretization of an underlying linear ill-posed problem. Our focus is on iterative regularization, and in particular the minimum-residual methods GMRES and MINRES, for which the projection onto the underlying Krylov subspace may have a regularizing effect (and the dimension of the Krylov subspace therefore acts as a regularization parameter). Our goal is to study some of the mechanisms behind this behavior.

The singular value decomposition (SVD) $A = U\Sigma V^T = \sum_{i=1}^{n} u_i \sigma_i v_i^T$ provides a natural tool for analysis of discrete ill-posed problems, for which the singular values $\sigma_i$ cluster at zero, and the right-hand side coefficients $u_i^T b$ satisfy the discrete Picard condition (DPC): on average, their absolute values decay faster than the singular values.

In the presence of noise in the right-hand side $b$, the "naive" solution $A^{-1}b$ is completely dominated by inverted noise. Regularized solutions can be computed by truncating or filtering the SVD expansion. For example, the truncated SVD (TSVD) method yields solutions $x_k = \sum_{i=1}^{k} \sigma_i^{-1}(u_i^T b)\, v_i$. Tikhonov regularization is another well-known method which, in its standard form, takes the

*Received xxx. Revised xxx. Communicated by xxx.

form $x_\lambda = \operatorname{argmin}_x \left\{ \|b - Ax\|_2^2 + \lambda^2 \|x\|_2^2 \right\}$, and the solution $x_\lambda$ can be written in terms of the SVD of $A$ as $x_\lambda = \sum_{i=1}^{n} \sigma_i (\sigma_i^2 + \lambda^2)^{-1} (u_i^T b)\, v_i$.

In general, a filtered SVD solution takes the form

$$x_{\text{filt}} = \sum_{i=1}^{n} \phi_i \frac{u_i^T b}{\sigma_i} v_i = V \Phi \Sigma^\dagger U^T b, \tag{1.1}$$

where $\Phi = \operatorname{diag}(\phi_i)$ is a diagonal filter matrix, and the filter factors are $\phi_i \in \{0, 1\}$ for TSVD and $\phi_i = \sigma_i^2/(\sigma_i^2 + \lambda^2)$ for Tikhonov regularization. Other regularization methods take a similar form, with different expressions for the filter factors. The effect of the filter is to remove the SVD components corresponding to the smaller singular values, and thereby to stabilize the solution.

The TSVD and Tikhonov methods are not always suited for large-scale problems. An alternative is to use *iterative regularization*, i.e., to apply an iterative method directly to $A\,x = b$ or $\min_x \|b - A\,x\|_2$ and obtain a regularized solution by early termination of the iterations. These methods exhibit semi-convergence, which means that the iterative solution improves during the first iterations, while at later stages the inverted noise starts to deteriorate the solution. For example, CGLS – which implicitly applies conjugate gradients to the normal equations $A^T Ax = A^T b$ – has this desired effect [8], [9], [14].

Other minimum-residual methods have also attained interest as iterative regularization methods. For some problems with a symmetric $A$, the algorithms MINRES [16] and MR-II [8] (which avoid the implicit cross-product $A^T A$ in CGLS) have favorable properties [8], [10], [14], [15]; in other situations they converge slower than CGLS.

If $A$ is nonsymmetric and multiplication with $A^T$ is difficult or impractical to compute, then CGLS is not applicable. GMRES [17] may seem as a natural candidate method for such problems, but only a few attempts have been made to investigate the regularization properties of this method and its variant RRGMRES [3], cf. [2], [4], [5].

The goal of this work is to perform a systematic study of the regularization properties of GMRES and related minimum-residual methods for discrete ill-posed problems, similar to Hanke's study [9] of the regularization properties of CGLS. Our focus is on the underlying mechanisms, and we seek to explain why – and when – such methods can be used for regularization. The hope is that our analysis will give better intuitive insight into the mechanisms of the regularization properties of these methods, which can aid the user in the choice of method.

In §2 we outline the theory for the minimum-residual methods considered here, and in §3 we take a closer look at the basis vectors for the underlying Krylov subspaces. In §4 we perform a theoretical and experimental study of the behavior of MINRES and the variant MR-II applied to symmetric indefinite problems, and in §5 we perform a similar analysis of GMRES and the variant RRGMRES applied to nonsymmetric problems.

Table 2.1: Minimum-residual methods and their Krylov subspaces.

| **Matrix** | **Algorithm** | **Krylov subspace** | **Solution** |
|---|---|---|---|
| Symmetric | MINRES | $\mathcal{K}_k(A, b)$ | $x^{(k)}$ |
| | MR-II | $\mathcal{K}_k(A, Ab)$ | $\bar{x}^{(k)}$ |
| Nonsymmetric and square | GMRES | $\mathcal{K}_k(A, b)$ | $x^{(k)}$ |
| | RRGMRES | $\mathcal{K}_k(A, Ab)$ | $\bar{x}^{(k)}$ |
| Any | CGLS, LSQR | $\mathcal{K}_k(A^T A, A^T b)$ | $\hat{x}^{(k)}$ |

## 2 Minimum-Residual Krylov Subspace Methods

In a projection method we seek an approximate solution to $A\,x = b$ for $x \in \mathcal{S}_k$, where $\mathcal{S}_k$ is some suitable $k$-dimensional subspace. Minimum-residual methods are special projection methods where the criterion for choosing $x$ amounts to minimization of the 2-norm of the residual:

$$\min_x \|b - A\,x\|_2, \qquad \text{s.t.} \qquad x \in \mathcal{S}_k.$$

For example, the TSVD solution $x_k$ minimizes the 2-norm of the residual over the subspace $\mathcal{S}_k = \text{span}\{v_1, v_2, \ldots, v_k\}$ spanned by the first $k$ right singular vectors. For the solution subspace $\mathcal{S}_k$ we can also use a Krylov subspace, such as $\mathcal{K}_k(A, b) \equiv \text{span}\{b, Ab, \ldots, A^{k-1}b\}$. Table 2.1 lists several Krylov subspace methods, and we note that only CGLS allows a rectangular $A$ matrix.

The variants MINRES and MR-II for symmetric matrices are based on three-term recurrence schemes for generating the desired Krylov subspaces, while GMRES and RRGMRES need to carry along the entire set of basis vectors for their Krylov subspaces. Throughout this paper, we always use the symmetric variants when $A$ is symmetric.

The methods RRGMRES and MR-II based on $\mathcal{K}_k(A, Ab)$ were originally designed for solving singular and inconsistent systems, and they restrict the Krylov subspace to be a subspace of $\mathcal{R}(A)$, the range of $A$. When $A = A^T$ this has the effect that MR-II computes the minimum-norm least squares solution [7]; for a general matrix $A$, RRGMRES computes the minimum-norm least squares solution when $\mathcal{R}(A) = \mathcal{R}(A^T)$ [3]. This is not the case for MINRES and GMRES.

When solving discrete ill-posed problems, we are not interested in the final convergence to the minimum-norm least squares solution, but rather in a good regularized solution. The Krylov subspace $\mathcal{K}_k(A, A\,b)$ may still be favorable to $\mathcal{K}_k(A, b)$, because the noise in the initial Krylov vector of the former subspace is damped by multiplication with $A$. A similar effect is automatically achieved in the CGLS subspace $\mathcal{K}_k(A^T A, A^T b)$ due to the starting vector $A^T b$. The subspace $\mathcal{K}_k(A, b)$ includes directly the noise component present in $b$, which can have a dramatic and undesirable influence on the early iteration vectors. For

this reason, RRGMRES and MR-II may provide better regularized solutions than GMRES and MINRES.

Our analysis of the algorithms is based on the fact that any solution in a Krylov subspace can be written in polynomial form. For example, for GMRES or MINRES we can write the $k$th iterate as

$$x^{(k)} = \mathcal{P}_k(A)\, b,$$

where $\mathcal{P}_k$ is a polynomial of degree $\leq k-1$. The corresponding residual $b - A\, x^{(k)}$ is therefore given by

$$b - A\, \mathcal{P}_k(A)\, b = \big(I - A\, \mathcal{P}_k(A)\big)\, b = \mathcal{Q}_k(A)\, b,$$

where $\mathcal{Q}_k(A) = I - A\, \mathcal{P}_k(A)$ is a polynomial of degree $\leq k$ with $\mathcal{Q}_k(0) = 1$. There are similar expressions for the RRGMRES/MR-II and CGLS solutions:

$$\bar{x}^{(k)} = \overline{\mathcal{P}}_{k+1}(A)\, b, \qquad \hat{x}^{(k)} = \widehat{\mathcal{P}}_k(A^T A)\, A^T b.$$

The RRGMRES/MR-II polynomial $\overline{\mathcal{P}}_{k+1}$ has degree $\leq k$ (instead of $k-1$) and the constant term is zero by definition.

The SVD of $A$ allows us to carry out a more careful study of the Krylov subspaces. For the GMRES and RRGMRES Krylov subspaces we obtain

$$\begin{aligned} \mathcal{K}_k(A, b) &= \text{span}\{b, U\Sigma V^T b, \ldots, (U\Sigma V^T)^{k-1} b\}, \\ \mathcal{K}_k(A, Ab) &= \text{span}\{U\Sigma V^T b, (U\Sigma V^T)^2 b, \ldots, (U\Sigma V^T)^{k} b\}, \end{aligned}$$

respectively. If we define the orthogonal matrix $C$ as well as the vector $\beta$ by

$$C = V^T U, \qquad \beta = U^T b, \tag{2.1}$$

then the GMRES iterates $x^{(k)}$ and the RRGMRES iterates $\bar{x}^{(k)}$ satisfy

$$V^T x^{(k)} \in \mathcal{K}_k(C\Sigma, C\beta), \qquad V^T \bar{x}^{(k)} \in \mathcal{K}_k(C\Sigma, C\Sigma\, C\beta). \tag{2.2}$$

It follows that we can write the GMRES and RRGMRES solutions as

$$x^{(k)} = V\, \Phi_k\, \Sigma^\dagger \beta, \qquad \Phi_k = \mathcal{P}_k(C\,\Sigma) C\,\Sigma, \tag{2.3}$$

$$\bar{x}^{(k)} = V\, \overline{\Phi}_k\, \Sigma^\dagger \beta, \qquad \overline{\Phi}_k = \overline{\mathcal{P}}_{k+1}(C\,\Sigma) C\,\Sigma. \tag{2.4}$$

Due to the presence of the matrix $C$, the "filter matrices" $\Phi_k$ and $\overline{\Phi}_k$ are full, in general. Hence, neither the GMRES nor the RRGMRES iterates have a filtered SVD expansion of the form (1.1) (the SVD components are "mixed" in each iteration), and therefore we cannot expect that these iterates resemble the TSVD or Tikhonov solutions.

When $A$ is symmetric we can write $A = V\, \Omega\Sigma\, V^T$, where $\Omega = \text{diag}(\pm 1)$ is a signature matrix and $\Omega\Sigma$ contains the eigenvalues of $A$. Hence $C = \Omega$, and the Krylov subspaces in (2.2) simplify to

$$V^T x^{(k)} \in \mathcal{K}_k\big(\Omega\Sigma, \Omega\beta\big), \qquad V^T \bar{x}^{(k)} \in \mathcal{K}_k(\Omega\Sigma, \Sigma\beta). \tag{2.5}$$

In this case the "filter matrices" $\Phi_k = \mathcal{P}_k(\Omega\Sigma)\Omega\Sigma$ and $\overline{\Phi}_k = \overline{\mathcal{P}}_{k+1}(\Omega\Sigma)\Omega\Sigma$ are diagonal (possibly with some negative elements), and therefore the MINRES and MR-II iterates have simple expressions in the SVD basis.

For the CGLS algorithm we have $A^TA = V\Sigma^2V^T$, and it follows that

$$\mathcal{K}_k(A^TA, A^Tb) = \text{span}\{V\Sigma U^Tb, V\Sigma^3U^Tb, \ldots, V\Sigma^{2k-1}U^Tb\}, \tag{2.6}$$

and that the CGLS iterates $\hat{x}^{(k)}$ satisfy

$$\hat{x}^{(k)} = V\,\widehat{\Phi}_k\,\Sigma^\dagger\beta, \qquad \widehat{\Phi}_k = \widehat{\mathcal{P}}_k(\Sigma^2)\Sigma^2, \tag{2.7}$$

where $\widehat{\Phi}_k$ is a diagonal matrix and $\widehat{\mathcal{P}}_k$ is the CGLS polynomial introduced above. This relation shows that the CGLS iterates $\hat{x}^{(k)}$ also have a simple expression in the SVD basis, namely, as a filtered SVD expansion of the form (1.1) with nonnegative filter factors given in terms of the polynomial $\widehat{\mathcal{P}}_k$.

The Krylov vectors for CGLS, MINRES and MR-II, respectively, have the form

$$V\Sigma^{2k-1}\beta, \qquad V(\Omega\Sigma)^{k-1}\Omega\beta \qquad \text{and} \qquad V(\Omega\Sigma)^{k-1}\Sigma\beta, \tag{2.8}$$

for $k = 1, \ldots, n$. The diagonal elements of $\Sigma$ decay, and so do the coefficients in $\beta$, on average, due to the DPC. Therefore the first SVD components will, in general, be more strongly represented in these Krylov subspaces than SVD components corresponding to smaller singular values. This indicates a correspondence of the CGLS, MINRES and MR-II solutions with both TSVD and Tikhonov solutions which are also, primarily, spanned by the first right singular vectors. A similar argument cannot be used to demonstrate that the GMRES and RRGMRES solutions are regularized solutions, due to the mixing by the full matrix $C$ in each iteration.

## 3 A Closer Look at the Solution Subspaces

Depending on the method used, the iterates lie in one of the three subspaces $\mathcal{R}(A)$, $\mathcal{R}([A, b])$ and $\mathcal{R}(A^T)$, as listed in Table 3.1, and this has an effect on the regularized solutions produced by the three methods. For this study, it is important to note that discrete ill-posed problems *in practise* – due to the decaying singular values and the effects of finite-precision arithmetic – behave as if the matrix $A$ is *rank deficient*.

From Table 3.1 it is clear why CGLS and MR-II are successful as iterative regularization methods: they produce solutions in subspaces of $\mathcal{R}(A^T)$, similar to TSVD and Tikhonov. MINRES can also be used, but we get a subspace of $\mathcal{R}(A^T)$ only for consistent systems – and unfortunately discrete ill-posed problems with noisy data behave as inconsistent problems. Neither GMRES nor RRGMRES produce solutions in the desired subspace $\mathcal{R}(A^T)$.

For symmetric matrices, the DPC implies that all the Krylov vectors in Eq. (2.8) have elements which, on average, decay for increasing index $i$. However, due to the different powers of $\Sigma$, the damping imposed by the multiplication with the

Table 3.1: Overview of the fundamental subspaces in which the iterates lie, with $\mathcal{R}(A) = \text{span}\{u_1, \ldots, u_r\}$, $\mathcal{R}([A, b]) = \text{span}\{b, u_1, \ldots, u_r\}$, $\mathcal{R}(A^T) = \text{span}\{v_1, \ldots, v_r\}$ and $r$ = rank of $A$.

| **Subspace** | **Method** | **Krylov subspace** | **Kind of system** |
|---|---|---|---|
| $\mathcal{R}(A)$ | GMRES | $\mathcal{K}_k(A, b)$ | consistent systems |
| | RRGMRES | $\mathcal{K}_k(A, Ab)$ | all systems |
| $\mathcal{R}([A, b])$ | GMRES | $\mathcal{K}_k(A, b)$ | inconsistent systems |
| | MINRES | $\mathcal{K}_k(A, b)$ | inconsistent systems |
| $\mathcal{R}(A^T)$ | MINRES | $\mathcal{K}_k(A, b)$ | consistent systems |
| | MR-II | $\mathcal{K}_k(A, Ab)$ | all systems |
| | CGLS | $\mathcal{K}_k(A^TA, A^Tb)$ | all systems |

singular values is different for these methods. For example, the $k$th CGLS Krylov vector is equal to the $2k$th Krylov vector of MINRES and the $(2k-1)$st Krylov vector of MR-II. Moreover, the vector $b$ appears undamped in the MINRES basis, while in the CGLS and MR-II bases it is always damped by $A^T$ and $A$, respectively. In the presence of noise, the fact that $b$ appears undamped in the MINRES Krylov subspace $\mathcal{R}([A, b]) = \mathcal{R}([A^T, b])$ can have a dramatic impact, as we illustrate below.

For nonsymmetric matrices the behavior of CGLS is identical to the symmetric case. On the other hand, the Krylov vectors for GMRES and RRGMRES are different, cf. (2.2); even if the DPC is fulfilled, we cannot be sure that the coefficients $|v_i^T b|$ decay, on average, for increasing index $i$. Furthermore, due to the presence of the non-diagonal matrix $C\Sigma$, no structured damping of these coefficients is obtained because the SVD components are "mixed." This means that GMRES and RRGMRES, in general, cannot be assumed to produce a solution subspace that resembles that spanned by the first right singular vectors.

Below we illustrate these issues with two numerical examples, in which the $n \times k$ matrices $W_k$, $\overline{W}_k$ and $\widehat{W}_k$ have orthonormal columns that span the Krylov subspaces for GMRES/MINRES, RRGMRES/MR-II and CGLS, respectively.

### *3.1 Example: Krylov Subspaces for a Symmetric Matrix*

We use the symmetric problem deriv2(100,3) from Regularization Tools [11] with a $100 \times 100$ coefficient matrix, and add white Gaussian noise $e$ to the right-hand side such that $b = A\,x_{\text{exact}} + e$ with $\|e\|_2 / \|Ax_{\text{exact}}\|_2 = 5 \cdot 10^{-4}$. Figure 3.1 shows the relative errors for a series of CGLS, MINRES and MR-II iterates, as well as the relative errors of similar TSVD solutions.

MINRES does not reduce the relative error as much as the other methods due to the noise component in the initial Krylov vector. MR-II and CGLS reduce the

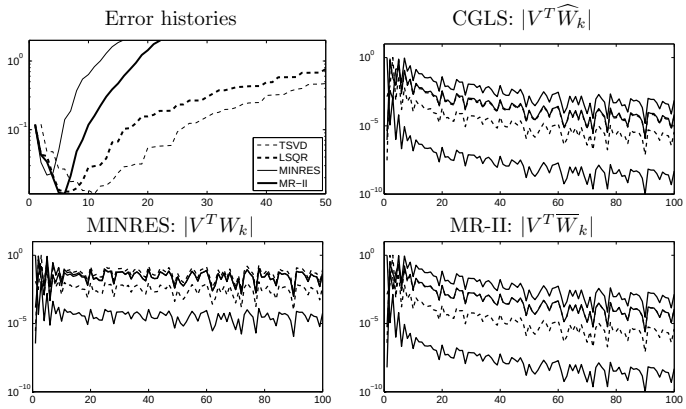



Figure 3.1: Symmetric test problem deriv2(100,3). Top left: the relative error $\|x_{\text{exact}} - x^{(k)}\|_2/\|x_{\text{exact}}\|_2$ for CGLS, MINRES and MR-II, and $\|x_{\text{exact}} - x_k\|_2/\|x_{\text{exact}}\|_2$ for TSVD. Remaining plots: the first five orthonormal Krylov vectors of the CGLS, MINRES and MR-II subspaces in the SVD basis.

relative error to about the same level, 0.0117 for MR-II and 0.0125 for CGLS, in 5–6 iterations. This makes MR-II favorable for this problem, because the number of matrix-vector multiplications is halved compared to CGLS. The best TSVD solution includes 11 SVD components, which indicates that the CGLS and MR-II solution subspaces are superior compared to the TSVD solution subspace of equal dimensions. This fact was originally noticed by Hanke [9].

Figure 3.1 also shows the first five orthonormal Krylov vectors expressed in terms of the right singular vectors $v_i$, i.e., the first five columns of $|V^TW_k|$, $|V^T\overline{W}_k|$ and $|V^T\widehat{W}_k|$. We see that the CGLS and MR-II vectors are mainly spanned by the first right singular vectors as expected, and that the contribution from the latter singular vectors is damped. We also see that the contribution from the latter right singular vectors is much more pronounced for MINRES due to the direct inclusion of the noise.

### *3.2 Example: Krylov Subspaces for a Nonsymmetric Matrix*

Here we use the nonsymmetric problem ilaplace(100) from Regularization Tools [11] with a coefficient matrix of size $100 \times 100$ and additive white Gaussian noise $e$ with $\|e\|_2/\|Ax_{\text{exact}}\|_2 = 5 \cdot 10^{-4}$. Figure 3.2 shows plots similar to those for the symmetric case. Only TSVD and CGLS are able to reduce the relative error considerably; neither GMRES nor RRGMRES produce iterates with small relative errors.

We note that the CGLS Krylov vectors behave similar to the symmetric case, i.e., the components that correspond to small singular values are effectively

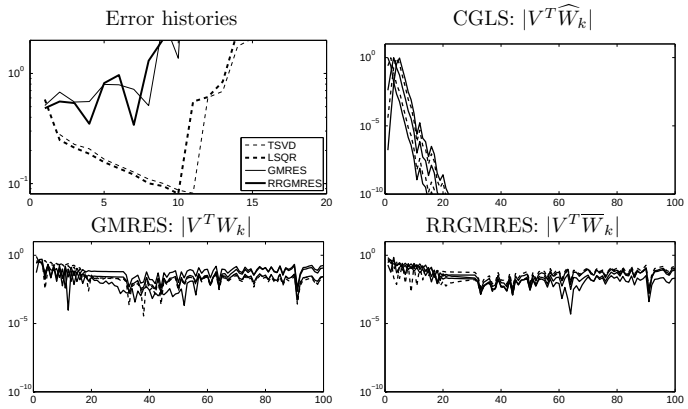



Figure 3.2: Nonsymmetric test problem ilaplace(100). Top left: the relative errors $\|x_{\text{exact}} - x^{(k)}\|_2/\|x_{\text{exact}}\|_2$ for CGLS, GMRES and RRGMRES, and $\|x_{\text{exact}} - x_k\|_2/\|x_{\text{exact}}\|_2$ for TSVD. Remaining plots: the first five orthonormal Krylov vectors of the CGLS, GMRES and RRGMRES subspaces in the SVD basis.

damped. Furthermore, we see that the GMRES and RRGMRES Krylov vectors do not exhibit any particular damping. In fact, all Krylov vectors contain significant components along all right singular vectors – including those that correspond to small singular values. Therefore the GMRES and RRGMRES iterates are composed not only of the first right singular vectors, they also include significant components in the direction of the last right singular vectors.

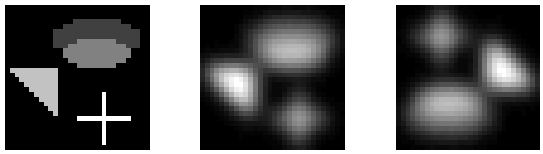
## 4 Iterative Regularization with MINRES and MR-II

We can express the MINRES and MR-II residual norms as

$$\|b - A\,x^{(k)}\|_2 = \|\mathcal{Q}_k(\Omega\Sigma)\,\beta\|_2, \qquad \|b - A\,\bar{x}^{(k)}\|_2 = \|\overline{\mathcal{Q}}_{k+1}(\Omega\Sigma)\,\beta\|_2,$$

where $\mathcal{Q}_k$ is the MINRES residual polynomial defined in Section 2, and $\overline{\mathcal{Q}}_{k+1} = I - A\,\overline{\mathcal{P}}_{k+1}(A)$ is the MR-II residual polynomial. Since these methods minimize the residual's 2-norm in each iteration, the effect of the residual polynomial is to "kill" the large components of $|\beta|$. Hence the residual polynomials must be small at those eigenvalues for which $|\beta|$ has large elements. (On the other hand, if a component $|\beta|$ is small, then the corresponding value of the residual polynomial need not be small.) Thus, MINRES and MR-II have the same intrinsic regularization property as the CGLS algorithm.

The main difference is that CGLS only has to "kill" components for positive eigenvalues of $A^TA$, while – for indefinite matrices – MINRES and MR-II must "kill" components corresponding to both positive and negative eigenvalues of $A$. The latter is more difficult, due to the polynomial constraints



$x = \mathrm{vec}(X)$ $b = \mathrm{vec}(B)$ $Pb$

Figure 4.1: True and blurred images for the symmetric two-dimensional problem.

$\mathcal{Q}_k(0) = \overline{\mathcal{Q}}_{k+1}(0) = 1$. These issues have been studied in more detail for MINRES [14], [15], and also for general symmetric minimum-residual methods, see [6] and the references therein.

Below we present two examples which illustrate the above observations and support the results obtained by Kilmer and Stewart [15]. Our examples also illustrate that the definiteness of the coefficient matrix affects the convergence and the iterates of the MINRES and MR-II, while both methods still produce regularized solutions. Furthermore, we demonstrate that MINRES and MR-II concentrate on the components that are significant for reducing the residual, i.e., the large right-hand side components in the eigenvector basis. A similar observation was done by Hanke [9] for CGLS.

### *4.1 Example: Image Deblurring with a Symmetric Matrix*

Let $X \in \mathbb{R}^{30\times 30}$ be the sharp image seen in Fig. 4.1 (left), and let the matrix $A$ be a discretization of a two-dimensional Gaussian blurring [12] with zero boundary conditions. The blurred image $B \in \mathbb{R}^{30\times 30}$ is shown in Fig. 4.1 (middle). The discrete ill-posed problem takes the form $A\,x = b$ where $x = \mathrm{vec}(X)$ is the column-wise stacked image $X$, and $b = \mathrm{vec}(B)$ is the stacked image $B$. We added white Gaussian noise to $b$ with $\|e\|_2/\|A\,x_{\mathrm{exact}}\|_2 = 10^{-2}$.

The coefficient matrix $A$ is both symmetric and persymmetric, i.e., $A = A^T$ and $PA = (PA)^T$, where $P$ is the reversal matrix. Moreover, $A$ is positive definite and $PA$ is indefinite. Since the 2-norm is invariant under multiplication with $P$ it follows that $\|b - A\,x\|_2 = \|P\,b - PA\,x\|_2$ where the permuted vector $Pb$ represents a 180° rotation of $B$ as shown in Fig. 4.1 (right). We apply CGLS, MINRES and MR-II to the two problems

$$A\,x = b \qquad \text{and} \qquad PA\,x = P\,b. \tag{4.1}$$

Obviously, the CGLS Krylov subspaces for the two problems are identical because $\mathcal{K}_k(A^TP^TPA, A^TP^TPb) = \mathcal{K}_k(A^TA, A^TA)$. However, this is not the case for MINRES and MR-II because $\mathcal{K}_k(A, b) \neq \mathcal{K}_k(PA, Pb)$ and $\mathcal{K}_k(A, Ab) \neq \mathcal{K}_k(PA, PAPb)$.

Figure 4.2 shows the reduction of the residual norms for the three methods applied to both versions of the problem. No reorthogonalization of the Krylov vectors is performed. Obviously, the convergence of CGLS is the same for the

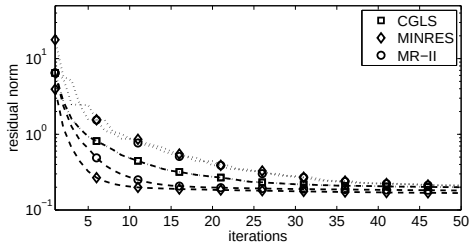



Figure 4.2: Reduction of the residual for CGLS, MINRES and MR-II, for the original problem (dashed lines) and the permuted problem (dotted lines) in (4.1).

Table 4.1: Number of iterations and relative error for best iterates, for the deblurring problems in §4.1 and §4.2.

| | CGLS | | MINRES | | MR-II | |
|---|---|---|---|---|---|---|
| Problem | its | rel. err | its | rel. err | its | rel. err |
| $A\,x = b$ | 59 | 0.4680 | 6 | 0.4916 | 20 | 0.4678 |
| $PA\,x = P\,b$ | 59 | 0.4680 | 81 | 0.4682 | 79 | 0.4681 |
| $A\,x' = b'$ | 84 | 0.4823 | 9 | 0.5166 | 24 | 0.4817 |
| $PA\,x' = P\,b'$ | 83 | 0.4823 | 95 | 0.4864 | 93 | 0.4834 |

two problems, whereas the convergence of MINRES and MR-II is *faster* than the convergence of CGLS when applied to $Ax = b$, and *slower* when applied to the permuted problem.

Figure 4.3 shows the eigenvalues for the two problems, together with the residual polynomials for the first four iterations of MINRES and MR-II. Both residual polynomials satisfy $\mathcal{Q}_k(0) = 1$ and $\overline{\mathcal{Q}}_{k+1}(0) = 1$, and in addition $\overline{\mathcal{Q}}'_{k+1}(0) = 0$. We see that the residual polynomials behave better – i.e., they are small for a greater range of eigenvalues – when all eigenvalues are positive. This explains why the convergence is faster for the problem with the positive definite matrix $A$.

Table 4.1 gives more information about the convergence of the two methods for the two problems, namely, the number of iterations and the relative error of the iterates with the smallest relative errors (compared to the exact solution). CGLS performs identically for the two problems. For the original problem, CGLS and MR-II produce slightly better solutions than MINRES, and MR-II needs much fewer iterations than CGLS; moreover MINRES produces a slightly inferior solution in only six iterations. For the permuted problem all three methods produce solutions of the same quality. MR-II and MINRES are comparable in speed and faster than CGLS (because CGLS needs two matrix-vector products per iteration).

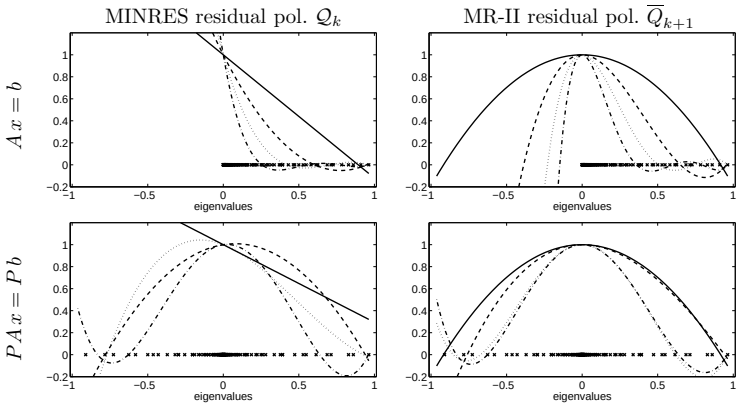
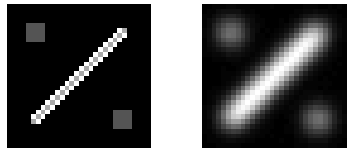



Figure 4.3: The first four residual polynomials (solid, dashed, dotted, and dash-dotted lines) for MINRES and MR-II applied to the original positive definite problem $A\,x = b$ and the permuted indefinite problem $PA\,x = P\,b$. The eigenvalues of $A$ and $PA$, respectively, are shown by the small crosses.

### *4.2 Example: The Role of the Solution Coefficients*

The residual polynomials of the methods also depend on the coefficients of the solution (and the right-hand side) in the eigenvector basis; not just the eigenvalues. To illustrate this, we create another sharp image $X'$ which is invariant to a 180° rotation, i.e., the column-wise stacked image satisfies $Px' = x'$. The symmetries of $A$ imply that the blurred image $B'$ also satisfies $Pb' = b'$, and again the noise level in $b$ is such that $\|e\|_2/\|Ax_{\text{exact}}\|_2 = 10^{-2}$. The sharp and blurred images are shown in Fig. 4.4.

For this particular right-hand side, the components in the eigenvector basis that correspond to negative eigenvalues of $PA$ are small (they are nonzero



$x' = \text{vec}(X')$ $b' = \text{vec}(B')$

Figure 4.4: True and blurred images for the modified problem $A\,x' = b'$.

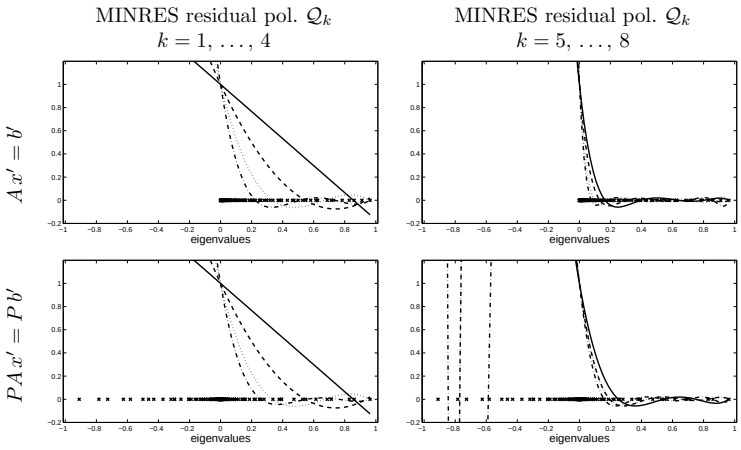



Figure 4.5: MINRES residual polynomials for the modified problems $A\,x' = b'$ and $PA\,x' = P\,b'$ (where $A$ is positive definite and $PA$ is indefinite). The eigenvalues are shown by the small crosses. The first four residual polynomials (bottom left plot) are not affected by the negative eigenvalues of $PA$, while the next four polynomials (bottom right plots) are.

solely due to the added noise). Therefore, in the initial iterations the residual polynomials need not pay as much attention to the negative eigenvalues. Figure 4.5 shows the MINRES residual polynomials corresponding to the first eight iterations, for both problems $A\,x' = b'$ and $PA\,x' = P\,b'$. Note that the first four polynomials are practically identical for the two problems, showing that MINRES is not affected by the small components corresponding to the negative eigenvalues. For the next four iterations, the small noise components for the negative eigenvalues start to affect the residual polynomials, thus slowing down the convergence. The situation is similar for the MR-II polynomials and not shown here.

This example illustrates that – at least in theory – the convergence for an indefinite problem can be similar to that of a definite problem. In practise, however, when noise is present in the right-hand side the convergence is always slower for the indefinite problem. Table 4.1 shows the convergence results which, in essence, are very similar to those for the previous example: both MINRES and MR-II produce regularized solutions, and in terms of computational work they are both favorable compared to CGLS.

## 5 Iterative Regularization with GMRES and RRGMRES

We now consider systems with nonsymmetric coefficient matrices and the Krylov methods GMRES and RRGMRES. Saad and Schultz [17, §3.4] showed that for any nonsingular matrix $A$ the GMRES iterations do not break down until the exact solution is found. On the other hand, as noted, e.g., by Brown and Walker [1], anything may happen when $A$ is singular. Our interest is in numerically singular systems (i.e., systems with tiny singular values), and we distinguish between rank-deficient problems and ill-posed problems.

### *5.1 Rank-Deficient Problems*

Rank-deficient problems are characterized by having a distinct gap between "large" and "small" singular values. If the underlying operator has a null space, then the matrix $A$ has small singular values whose size reflects the discretization scheme and the machine precision. Therefore, it makes sense to define the numerical subspaces $\mathcal{R}(A)$, $\mathcal{N}(A^T)$, $\mathcal{R}(A^T)$ and $\mathcal{N}(A)$, where the null spaces are spanned by the singular vectors corresponding to the small singular values.

If $A$ is rank-deficient and the noise in the right-hand side is so small that it primarily affects the SVD components outside $\mathcal{R}(A)$, then the minimum-norm least squares solution $A^\dagger b$ (which is really a TSVD solution) is a good regularized solution. On the other hand, if the noise in the right-hand side is larger such that it also affects the components of $b$ in $\mathcal{R}(A)$, then the problem effectively acts like a discrete ill-posed problem, and the solution must be further regularized to minimize the effect of the noise.

It was shown by Brown and Walker [1, Thm. 2.4] that GMRES computes the minimum-norm least squares solution if the system fulfills $\mathcal{N}(A) = \mathcal{N}(A^T)$ and if it is consistent. In this case GMRES constructs a solution in $\mathcal{R}(A) = \mathcal{R}(A^T)$, and it is obvious that if no solution components in $\mathcal{R}(A)$ are too affected by the noise, then GMRES will eventually produce a suitable regularized solution.

### *5.2 Example: Numerically Rank-Deficient Problem*

When the noise level is small, then the test problem `heat` from Regularization Tools [11] behaves as a numerically rank-deficient problem. For $n = 100$ the first 97 singular values lie between $10^{-7}$ and $10^{-3}$, while the last three are of the order $10^{-15}$. We add white Gaussian noise to $b$ with $\|e\|_2/\|Ax_{\text{exact}}\|_2 = 10^{-8}$, such that the first 97 SVD components of $b$ are almost unaffected by the noise.

Figure 5.1 shows the relative errors compared to the exact solution for GMRES, RRGMRES and CGLS, and we see that only CGLS converges. The reason is that for this problem we have $\mathcal{N}(A) = \text{span}\{v_{98}, v_{99}, v_{100}\} \neq \mathcal{N}(A^T) = \text{span}\{u_{98}, u_{99}, u_{100}\}$ (see Fig. 5.2), such that neither GMRES nor RRGMRES are guaranteed to produce the desired minimum-norm least squares solution.

### *5.3 Discrete Ill-Posed Problems*

For discrete ill-posed problems, the notion of numerical subspaces is not well defined due to decaying singular values with no gap in the spectrum. Moreover,

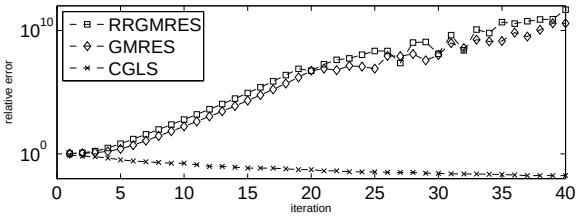



Figure 5.1: Relative errors for the first 40 iterations of GMRES, RRGMRES and CGLS applied to the rank-deficient inverse heat problem.

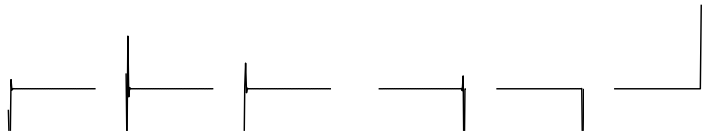


Figure 5.2: The vectors spanning $\mathcal{N}(A)$ and $\mathcal{N}(A^T)$ for the inverse heat problem. From left to right, the left null vectors $u_{98}$, $u_{99}$, $u_{100}$ and the right null vectors $v_{98}$, $v_{99}$, $v_{100}$.

for GMRES and RRGMRES a *mixing* of the SVD components occurs in each iteration, due to the presence of the matrix $C = V^T U$ in the expression for the Krylov vectors, cf. (2.2). The mixing takes place right from the start:

$$v_i^T b = \sum_{j=1}^{n} c_{ij}\, \beta_j, \qquad v_i^T A\, b = \sum_{j=1}^{n} \left( \sum_{\ell=1}^{n} \sigma_\ell\, c_{i\ell}\, c_{\ell j} \right) \beta_j \ .$$

An important observation here is that the noisy components of $b$ are also mixed by $C$, and therefore the non-diagonal filters of GMRES and RRGMRES not only change the solution subspaces, but also mix the contributions from the noisy components. This is contrary to spectral filtering methods such as TSVD, Tikhonov regularization, CGLS and MINRES/MR-II. In particular, the first iterations of GMRES and RRGMRES need not favor the less noisy SVD components in the solution, and the later iterations need not favor the more noisy components.

### *5.4 Example: Image Deblurring – GMRES and RRGMRES Work*

GMRES and RRGMRES were proposed for image deblurring in [2] and [4]; here we consider a test problem similar to Example 4.3 in [4] with spatially

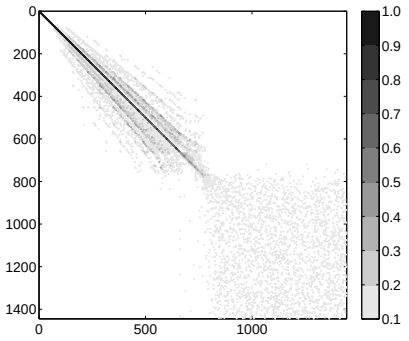



Figure 5.3: Illustration of the "structure" of the matrix $C = V^T U$ for the image deblurring problem; only matrix elements $|c_{ij}| \geq 0.1$ are shown.

variant Gaussian blur, in which the nonsymmetric coefficient matrix is given by

$$A = \begin{pmatrix} I_o & 0 \\ 0 & 0 \end{pmatrix} (T_1 \otimes T_1) + \begin{pmatrix} 0 & 0 \\ 0 & I_o \end{pmatrix} (T_2 \otimes T_2),$$

where $I_o$ is the $\frac{n}{2} \times \frac{n}{2}$ identity matrix, and $T_1$ and $T_2$ are $N \times N$ Toeplitz matrices:

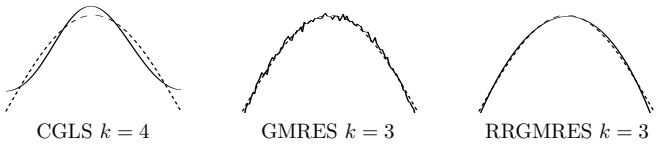
$$(T_\ell)_{ij} = \frac{1}{\sigma_\ell \sqrt{2\pi}} \exp\left( -\frac{1}{2} \left( \frac{i-j}{\sigma_\ell} \right)^2 \right), \qquad \ell = 1, 2$$

with $\sigma_1 = 4$ and $\sigma_2 = 4.5$. This models a situation where the left and right halves of the $N \times N$ image are degraded by two different point spread functions.

Here we use $N = 38$ (such that $n = N^2 = 1444$). For a problem of this size we can explicitly calculate the SVD of $A$. The singular values (not show here) decay gradually from $\sigma_1 = 1$ to $\sigma_{800} \approx 10^{-16}$, while the remaining singular values stay at this level. The "structure" of $C = V^T U$ is illustrated in Fig. 5.3 which shows all elements $|c_{ij}| \geq 0.1$. Although $A$ is nonsymmetric, the matrix $C$ is close to diagonal in the upper left corner (which corresponds to the numerically nonzero singular values). Therefore, GMRES and RRGMRES will only introduce a limited amount of "mixing" between the SVD components of the Krylov subspaces, and moreover the SVD components corresponding to the numerically zero singular values will not enter the Krylov subspaces. Hence GMRES and RRGMRES can be expected to compute regularized solutions in the SVD basis, confirming the experimental results in [4].

### *5.5 Example: Test Problem* baart *– GMRES and RRGMRES Work*

Here we use the test problem baart(100) from Regularization Tools [11], and we added white Gaussian noise $e$ to the right-hand side such that $\|e\|_2 / \|A\, x_{\text{exact}}\|_2 =$



CGLS $k = 4$ GMRES $k = 3$ RRGMRES $k = 3$

Figure 5.4: Best iterates of CGLS, GMRES and RRGMRES applied to the baart(100) test problem (the exact solution is shown by the dotted lines).

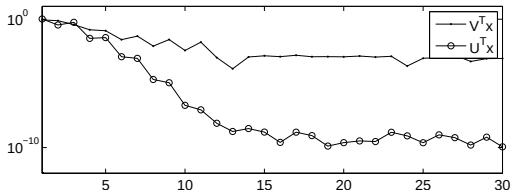



Figure 5.5: The coefficients of exact solution $x^{\text{exact}}$ to the baart(100) test problem in the bases of the left and right singular vectors.

$10^{-3}$. Figure 5.4 shows the best CGLS, GMRES and RRGMRES iterates. Note that especially RRGMRES approximates the exact solution very well; the best GMRES solution is somewhat more noisy, but it also approximates the exact solution quite well. The best CGLS solution is not noisy, but neither is it a good approximation to the exact solution.

For this problem, the matrix $C = V^T U$ (not shown here) is far from a diagonal matrix, and hence the SVD components are considerably mixed in the Krylov subspaces of GMRES and RRGMRES. The reason why these methods are able to approximate the exact solution so well is that the solution subspaces (spanned by the left singular vectors $u_i$) are favorable for approximating this particular solution.

Figure 5.5 shows the coefficients of the exact solution $x^{\text{exact}}$ in the left and right singular vector bases. The singular values $\sigma_i$ for $i > 15$ are at the rounding error level, and therefore the coefficients $u_i^T x^{\text{exact}}$ and $v_i^T x^{\text{exact}}$ are unreliable for $i > 15$. We see that $x^{\text{exact}}$ is indeed well expressed by a few left singular vectors $u_i$, while more right singular vectors $v_i$ (the "usual" basis vectors for regularized solutions) are needed. I.e., the solution we seek is better represented in a small subspace of $\mathcal{R}(A)$ than in a subspace of $\mathcal{R}(A^T)$, cf. Table 2.1. For this particular problem, both CGLS, TSVD and Tikhonov regularization are not particularly well suited, and GMRES and RRGMRES happen to produce better solution subspaces and better regularized solutions.

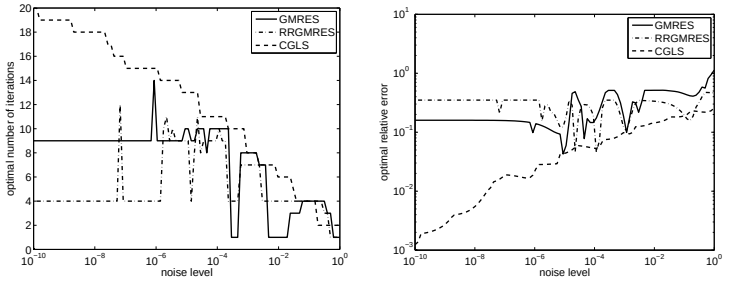



Figure 5.6: Results for the ilaplace(100) test problem. Left: the iteration count $k$ for optimal CGLS, GMRES and RRGMRES solutions as a function of the noise level. Right: the corresponding relative errors $\|x^{\text{exact}} - x^{(k)}\|_2 / \|x^{\text{exact}}\|_2$ versus the noise level.

### *5.6 Example: Test Problem ilaplace – GMRES and RRGMRES Fail*

Finally, we consider the nonsymmetric problem ilaplace(100) from Regularization Tools [11]. The singular values decay gradually and hit the machine precision around index 30. The noise vector $e$ contains white Gaussian noise, and we use 100 different scalings of $e$ such that $\|e\|_2 / \|Ax_{\text{exact}}\|_2$ is logarithmically distributed between $10^{-10}$ and $10^{0}$.

For each noise level, we perform 30 iterates of CGLS, GMRES and RRGMRES, and the iterates with smallest relative errors are found for all methods and all noise levels. Figure 5.6 shows the iteration count for the best solution as a function of the noise level, as well as the corresponding relative errors versus the noise level.

For CGLS there is a clear correspondence between the noise level and the number of iterations – for a high noise level only few iterations can be performed before the noise distorts the solutions, while more iterations can be performed for a lower noise level. The plot of the relative errors confirms that, overall, the solutions get better as the noise level decreases.

The same behavior is *not* observed for GMRES and RRGMRES. For small noise levels, the solutions do not improved beyond a certain level, and for higher noise levels the optimal number of iterations is unpredictable. For example, for the noise level $10^{-10}$, the 4th RRGMRES iterate is the optimal solution, with a relative error of 0.3500, but for the higher noise level $1.15 \cdot 10^{-4}$ the best RRGMRES iterate is the 10th, with a relative error of only 0.04615.

As mentioned in §5.3 the sensitivity to the noise is indeed very different for GMRES and RRGMRES than for CGLS. The behavior observed for this example indicates that it may be very difficult to find suitable stopping rules for GMRES and RRGMRES.

*5.7 Normal Matrices*

If $A$ is normal then we can write $A = VDV^T$, where $V$ is orthogonal and $D$ is block diagonal with $1 \times 1$ blocks $d_i$ and $2 \times 2$ blocks $D_i$ (see, e.g., [13, §2.5]). Here, $d_i = g_i\,\sigma_i$ with $g_i = \text{sign}(d_i)$ and $\sigma_i = |d_i|$, while

$$D_i = \begin{pmatrix} a_i & b_i \\ -b_i & a_i \end{pmatrix} = \sigma_i\, G_i, \qquad \sigma_i = \sqrt{a_i^2 + b_i^2} \qquad G_i = \begin{pmatrix} c_i & s_i \\ -s_i & c_i \end{pmatrix},$$

i.e., the $2 \times 2$ blocks are scaled Givens rotations with $c_i = a_i/\sigma_i$ and $s_i = b_i/\sigma_i$, and thus $D_i = G_i\,\Sigma_i$ with $\Sigma_i = \text{diag}(\sigma_i, \sigma_i)$.

Now collect all $g_i$ and $G_i$ in the orthogonal block diagonal matrix $G$, and all scaling factors $\sigma_i$ and $\Sigma_i$ in the diagonal matrix $\Sigma$. Then an SVD of $A$ is given by $A = U\Sigma V^T$ with $U = V\,G$, and it follows that $C = V^TU = G$. Therefore, for a normal matrix $A$, the mixing of the SVD components is limited to a mixing of subspaces of dimension two, and hence the Krylov subspaces for GMRES and RRGMRES are likely to be well behaved.

## 6 Conclusion

MINRES and MR-II have regularization properties for the same reason as CGLS does: by "killing" the large SVD components of the residual – in order to reduce its norm as much as possible – they capture the desired SVD components and produce a regularized solution. Negative eigenvalues do not inhibit the regularizing effect of MINRES and MR-II, but they influence the convergence rate.

GMRES and RRGMRES mix the SVD components in each iteration and thus do not provide a filtered SVD solution. For some problems GMRES and RRGMRES produce regularized solutions, either because the mixing is weak (see §5.4) or because the Krylov vectors are well suited for the problem (see §5.5). For other problems neither GMRES nor RRGMRES produce regularized solutions, either due to an unfavorable null space (see §5.2) or due to a severe and undesired mixing of the SVD components (see §5.6).

Our bottom-line conclusion is that while CGLS, MINRES and MR-II have general regularization properties, one should be very careful using GMRES and RRGMRES as general-purpose regularization methods for practical problems.

REFERENCES

1. P. N. Brown and H. F. Walker, *GMRES on (nearly) singular systems*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 37–51.
2. D. Calvetti, G. Landi, L. Reichel, and F. Sgallari, *Non-negativity and iterative methods for ill-posed problems*, Inverse Problems, 20 (2004), pp. 1747–1758.
3. D. Calvetti, B. Lewis, and L. Reichel, *GMRES-type methods for inconsistent systems*, Lin. Alg. Appl., 316 (2000), pp. 157–169.
4. D. Calvetti, B. Lewis, and L. Reichel, *GMRES, L-curves, and discrete ill-posed problems*, BIT, 42 (2002), pp. 44–65.

5. D. Calvetti, B. Lewis, and L. Reichel, *On the regularizing properties of the GMRES method*, Numer. Math., 91 (2002), pp. 605–625.
6. B. Fischer, *Polynomial Based Iteration Methods for Symmetric Linear Systems*, Wiley Teubner, Stuttgart, 1996.
7. B. Fischer, M. Hanke, and M. Hochbruck, *A note on conjugate-gradient type methods for indefinite and/or inconsistent linear systems*, Numer. Algo., 11 (1996) pp. 181–187.
8. M. Hanke, *Conjugate Gradient Type Methods for Ill-Posed Problems*, Longman Scientific & Technical, Essex, 1995.
9. M. Hanke, *On Lanczos based methods for the regularization of discrete ill-posed problems*, BIT, 41 (2001), pp. 1008–1018.
10. M. Hanke and J. G. Nagy, *Restoration of atmospherically blurred images by symmetric indefinite conjugate gradient techniques*, Inverse Problems, 12 (1996), pp. 157–173.
11. P. C. Hansen, *Regularization Tools. A Matlab package for analysis and solution of discrete ill-posed problems*, Numer. Algo., 6 (1004), pp. 1–35.
12. P. C. Hansen, J. G. Nagy, and D. P. O'Leary, *Deblurring Images – Matrices, Spectra and Filtering*, SIAM, Philadephia, 2006 (to appear).
13. R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 1985.
14. M. E. Kilmer, *On the regularizing properties of Krylov subspace methods*, unpublished; results presented at BIT 40th Anniversary meeting, Lund, Sweden, 2000.
15. M. E. Kilmer and G. W. Stewart, *Iterative regularization and MINRES*, SIAM J. Matrix Anal. Appl., 21 (1999), pp. 613–628.
16. C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Num. Anal., 12 (1975), pp. 617-629.
17. Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

# Bibliography


[1] anvari.org. Math jokes. http://www.anvari.org/shortjoke/Math/430.html.

[2] C. T. H. Baker. *The Numerical Treatment of Integral Equations*. Clarendon Press, Oxford, 1977.

[3] M. Belge, M. E. Kilmer, and E. L. Miller. Efficient determination of multiple regularization parameters in a generalized L-curve framework. *Inverse Problems*, 18:1161–1183, 2002.

[4] R. J. Blakely. *Potential Theory in Gravity and Magnetic Applications*. Cambridge University Press, 1996.

[5] P. N. Brown and H. F. Walker. GMRES on (nearly) singular systems. *SIAM J. Matrix Anal. Appl.*, 18:37–51, 1997.

[6] D. Calvetti, G. Landi, L. Reichel, and F. Sgallari. Non-negativity and iterative methods for ill-posed problems. *Inverse Problems*, 20:1747–1758, 2004.

[7] D. Calvetti, B. Lewis, and L. Reichel. GMRES-type methods for inconsistent systems. *Lin. Alg. Appl.*, 316:157–169, 2000.

[8] D. Calvetti, B. Lewis, and L. Reichel. On the choice of subspace for iterative methods for lilnear discrete ill-posed probblems. *Int. J. Appl. Math. Comput. Sci.*, 11:1069–1092, 2001.

[9] D. Calvetti, B. Lewis, and L. Reichel. GMRES, L-curves, and discrete ill-posed problems. *BIT*, 42:44–65, 2002.

[10] D. Calvetti, B. Lewis, and L. Reichel. A hybrid GMRES and TV-norm based method for image restoration. *Proceedings of SPIE*, 4791:192–200, 2002.

[11] D. Calvetti, B. Lewis, and L. Reichel. On the regularizing properties of the GMRES method. *Num. Math.*, 91:605–625, 2002.

[12] D. Calvetti, L. Reichel, and A. Shuibi. Smoothing preconditioners for ill-posed problems. Earlier version of [13].

[13] D. Calvetti, L. Reichel, and A. Shuibi. Invertible smoothing preconditioners for linear discrete ill-posed problems. *Appl. Num. Math.*, 54:135–149, 2005.

[14] D. Calvetti and E. Somersalo. Priorconditioners for linear systems. *Inverse Problems*, 21:1397–1418, 2005.

[15] A. Carasso. Determining surface temperatures from interior observations. *SIAM J. Appl. Math.*, 42:558–574, 1982.

[16] J. L. Castellanos, S. Gómez, and V. Guerra. The triangle method for finding the corner of the L-curve. *Appl. Num. Math.*, 43:359–373, 2002.

[17] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Trans. Img. Proc.*, 6:298–311, 1997.

[18] M. Donatelli, C. Estatico, J. Nagy, L. Perrone, and S. Serra-Capizzano. Anti-reflective boundary conditions and fast 2D deblurring models. *Proceedings of SPIE*, 5205:380–389, 2004.

[19] M. Donatelli and S. Serra-Capizzano. Anti-reflective boundary conditions and re-blurring. *Inverse Problems*, 21:169–182, 2005.

[20] M. Eiermann and O. G. Ernst. Geometric aspects of the theory of Krylov subspace methods. *Acta Num.*, 10:251–312, 2001.

[21] L. Eldén. A weighted pseudoinverse, generalized singular values, and constrained least squares problems. *BIT*, 22:487–502, 1982.

[22] H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems*. Kluwer Academic Publishers, Dordrecht, 2000.

[23] B. Fischer. *Polynomial Based Iteration Methods for Symmetric Linear Systems*. John Wiley & Sons Ltd. and B. G. Teuner, Stuttgart, 1996.

[24] B. Fischer, M. Hanke, and M. Hochbruck. A note on conjugate-gradient type methods for indefinite and/or inconsistent linear systems. *Num. Alg.*, 11:181–187, 1996.

[25] G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM J. Num. Anal.*, 2:205–224, 1965.

[26] G. H. Golub and C. F. van Loan. *Matrix Computations – third edition*. The John Hopkins University Press, Baltimore, 1996.

[27] S. Goossens and D. Roose. Ritz and harmonic Ritz values and the convergence of FOM and GMRES. *Num. Lin. Alg. Appl.*, 6:281–293, 1999.

[28] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. SIAM, Philadelphia, 1997.

[29] C. W. Groetsch. *The theory of Tikhonov regularization for Fredholm equations of the first kind*. Pitman, Boston, 1984.

[30] M. Gutknecht. *Iterative Methods*. Online – Part II of Software for Numerical Linear Algebra: http://www.sam.math.ethz.ch/~mhg/unt/SWNLA/SWNLA2.ps, 2005.

[31] M. Hanke. *Conjugate Gradient Type Methods for Ill-posed Problems*. Longman Scientific & Technical, Essex, 1995.

[32] M. Hanke. On Lanczos based methods for the regularization of discrete ill-posed problems. *BIT*, 41:1008–1018, 2001.

[33] M. Hanke and P. C. Hansen. Regularization methods for large-scale problems. *Surveys Math. Industry*, 3:253–315, 1993.

[34] M. Hanke and J. G. Nagy. Restoration of atmospherically blurred images by symmetric indefinite conjugate gradient techniques. *Inverse Problems*, 12:157–173, 1996.

[35] P. C. Hansen. The discrete Picard condition for discrete ill-posed problems. *BIT*, 30:658–672, 1990.

[36] P. C. Hansen. Regularization Tools – A MATLAB package for analysis and solution of discrete ill-posed problems. *Num. Alg.*, 6:1–35, 1994.

[37] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems*. SIAM, Philadelphia, 1998.

[38] P. C. Hansen. Deconvolution and regularization with Toeplitz matrices. *Num. Alg*, 29:323–378, 2002.

[39] P. C. Hansen. Oblique projections, pseudoinverse and standard-form transformation. Technical Report IMM-TR-2004-13, IMM, DTU, 2004.

[40] P. C. Hansen and T. K. Jensen. Smoothing-norm preconditioning for regularizing minimum-residual methods. *SIAM J. Matrix Anal. Appl.*, 2006.

[41] P. C. Hansen, T. K. Jensen, and G. Rodriguez. An adaptive pruning algorithm for the discrete L-curve criterion. *J. Comp. Appl. Math.*, 2006.

[42] P. C. Hansen, M. Kilmer, and R. H. Kjeldsen. Exploiting residual information in the parameter choice for discrete ill-posed problems. *BIT*, 46:41–59, 2006.

[43] P. C. Hansen, J. G. Nagy, and D. P. O'Leary. *Deblurring Images – Matrices, Spectra and Filtering*. SIAM, Philadephia, 2006 (to appear).

[44] P. C. Hansen and D. P. O'Leary. The use of the L-curve in the regularization of discrete ill-posed problems. *SIAM J. Sci. Comput.*, 14:1487–1503, 1993.

[45] G. Heinig and K. Rost. Representations of Toeplitz-plus-Hankel matrices using trigonometric transformations with application to fast matrix-vector multiplication. *Lin. Alg. Appl.*, 275-276:225–248, 1998.

[46] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[47] I. C. F. Ipsen. Expressions and bounds for the GMRES residual. *BIT*, 40:524–535, 2000.

[48] I. C. F. Ipsen and C. D. Meyer. The idea behind Krylov methods. *American Math. Monthly*, 105:889–899, 1998.

[49] M. Jacobsen. M$\mathcal{OO}$Re Tools – MATLAB Object-Oriented Regularization Tools. `http://www2.imm.dtu.dk/~tkj/MOOReTools`.

[50] M. Jacobsen. *Modular Regularization Algorithms*. PhD thesis, IMM-PHD-2004-140, Technical University of Denmark, 2004.

[51] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, Englewood Cliffs, 1989.

[52] T. K. Jensen. Regularizing iterations for image restorations. Master's thesis, IMM-THESIS-2003-18, Technical University of Denmark, 2003.

[53] T. K. Jensen and P. C. Hansen. Iterative regularization with minimum-residual methods. *Submitted to BIT*, 2006.

[54] J. Kamm and J. G. Nagy. Optimal kronecker product approximations of block Toeplitz matrices. *SIAM J. Matrix Anal. Appl.*, 22:155–172, 2000.

[55] N. B. Karayiannis and A. N. Venetsanopoulos. Regularization theory in image restoration – the stabilizing functional approach. *IEEE Trans. on Acoustics, Speech, and Signal Proc.*, 38:1155–1179, 1990.

[56] M. Kilmer and G. W. Steward. Iterative regularization and MINRES. *SIAM J. Matrix Anal. Appl.*, 21:613–628, 1999.

[57] M. E. Kilmer. *Regularization of Ill-posed Problems*. PhD thesis, University of Maryland at College Park, 1997.

[58] M. E. Kilmer. On the regularization properties of Krylov subspace methods. unpublished manuscript – http://www.tufts.edu/~mkilme01/, 2001.

[59] M. E. Kilmer and D. P. O'Leary. Computing regularization parameters in iterative methods for ill-posed problems. *SIAM J. Matrix Anal. Appl.*, 22:1204–1221, 2001.

[60] S. L. Lee. Best available bounds for departure from normality. *SIAM J. Matrix Anal. Appl.*, 17:984–991, 1996.

[61] J. Liesen. Computable convergence bounds for GMRES. *SIAM J. Matrix Anal. Appl.*, 21:882–903, 2000.

[62] J. Liesen and Z. Strakoš. Convergence of GMRES for tridiagonal Toeplitz matrices. *SIAM J. Matrix Anal.*, 26:233–251, 2004.

[63] J. Liesen and P. Tichý. Convergence analysis of Krylov subspace methods. *GAMM-Mitt.*, 27(2):153–173, 2004.

[64] J. Liesen and P. Tichý. The worst-case GMRES for normal matrices. *BIT*, 44:79–98, 2004.

[65] C. D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia, 2000.

[66] J. C. Meza. A modification to the GMRES method for ill-conditioned linear systems. Technical Report SAND95-8220, Sandia National Laboratories, 1995.

[67] A. F. J. Moffat. A theoretical investigation of focal stellar images in the photographic emulsion and application to photographic photometry. *Astron. and Astrophys.*, 3:455–461, 1969.

[68] V. A. Morozov. *Methods for Solving Incorrectly Posed Problems*. Springer-Verlag, 1984.

[69] J. G. Nagy, M. K. Ng, and L. Perrone. Kronecker product approximations for image restoration with reflexive boundary conditions. *SIAM J. Matrix Anal. Appl.*, 25:829–841, 2004.

[70] J. G. Nagy and D. P. O'Leary. Restoring images degraded by spatially variant blur. *SIAM J. Sci. Comput*, 19:1063–1082, 1998.

[71] J. G. Nagy, K. Palmer, and L. Perrone. Iterative methods for image deblurring: a Matlab object-oriented approach. *Num. Alg.*, 36:73–93, 2004.

[72] J. G. Nagy and Z. Strakos. Enforcing nonnegativity in image reconstruction algorithms. *Proceedings of SPIE*, 4121:182–190, 2000.

[73] A. S. Nemirovskii. The regularizing properties of the adjoint gradient method in ill-posed problems. *U.S.S.R. Comput. Maths. Math. Phys.*, 26:7–16, 1986.

[74] M. K. Ng, R. H. Chan, and W.-C. Tang. A fast algorithm for deblurring models with neumann boundary conditions. *SIAM J. Sci. Comput.*, 21:851–866, 1999.

[75] N. B. Nill. A visual model weighted cosine transform for image compression and quality assessment. *IEEE Trans. Com.*, COM-33:551–557, 1985.

[76] D. P. O'Leary. Robust regression computation using iteratively reweighted least squares. *SIAM J. Matrix Anal. Appl.*, 11:466–480, 1990.

[77] F. O'Sullivan. Discretized Laplacian smoothing by Fourier methods. *J. American Stat. Assoc.*, 86:634–642, 1991.

[78] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Num. Anal.*, 12:617–629, 1975.

[79] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, 8:43–71, 1982.

[80] B. N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice-Hall, Englewood Cliffs, N.J., 1980.

[81] L. C. Perrone. *Kronecker products in image restoration.* PhD thesis, Emory University, Atlanta, GA, 2004.

[82] M. Püschel and J. M. F. Moura. The algebraic approach to the discrete cosine and sine transforms and their fast algorithms. *SIAM J. Comput.*, 32:1280–1316, 2003.

[83] J. M. Rasmussen. Compact linear operators and Krylov subspace methods. Master's thesis, IMM-EKS-2001-9, Technical University of Denmark, 2001.

[84] G. Rodriguez and D. Theis. An algorithm for estimating the optimal regularization parameter by the L-curve. *Rend. di Matematica*, 25, 2005.

[85] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14:461–469, 1993.

[86] Y. Saad. *Iterative Methods for Sparse Linear Systems – Second Edition.* SIAM, Philadelphia, 2003.

[87] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[88] S. Serra-Capizzano. A note on antireflective boundary conditions and fast deblurring models. *SIAM J. Sci. Comput.*, 25:1307–1325, 2003.

[89] A. N. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Math. Dokl.*, 4:1035–1038, 1963.

[90] H. A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press, 2003.

[91] C. F. van Loan. The ubiquitous Kronecker product. *J. Comp. Appl. Math.*, 123:95–100, 2000.

[92] C. Vogel. Solving ill-conditioned linear systems using the conjugate gradient method. Technical report, Dept. Math. Science., Montana State Univ., 1987.

[93] C. R. Vogel. *Computational Methods for Inverse Problems*. SIAM Frontiers in Applied Mathematics, Philadelphia, 2002.

[94] I. Zavorin, D. P. O'leary, and H. Elman. Complete stagnation of GMRES. *Lin. Alg. and Appl.*, 367:165–183, 2003.